AD-A242 452



TASK-ANALYTIC DESIGN
OF
GRAPHIC PRESENTATIONS
Technical Report AIP - 145

Stephen Casner

The Artificial Intelligence and Psychology Project

DTIC
ELECTE
NOV 1 1991
S D

Departments of
Computer Science and Psychology
Carnegie Mellon University

Learning Research and Development Center
University of Pittsburgh

Approved for public release; distribution unlimited

TASK-ANALYTIC DESIGN OF GRAPHIC PRESENTATIONS

Technical Report AIP - 145

Stephen Casner

University of Pittsburgh
Pittsburgh, PA 15260

May 18, 1990

This research was supported by the Computer Sciences Division, Office of Naval Research, under Contract Number N00014-86-K-0678. Reproduction in whole or in part is permitted for purposes of the United States Government. Approved for public release; distribution unlimited.

91-14701

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release, distribution unlimited.
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) AIP-145	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION University of Pittsburgh 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION Computer Sciences Division Office of Naval Research (Code 1133)
6c ADDRESS (City, State, and ZIP Code) Department of Computer Science Pittsburgh, PA 15260	7b ADDRESS (City, State, and ZIP Code) 800 N. Quincy Street Arlington, VA 22217-5000
8a NAME OF FUNDING / SPONSORING ORGANIZATION Same as Monitoring Organization 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-86-K-0678
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
Task-analytic design of graphic presentations (unclassified)

12 PERSONAL AUTHOR(S)
Casner, Stephen Michael

13a TYPE OF REPORT	13b TIME COVERED FROM ____ TO ____	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
		1990, May 18	186

16 SUPPLEMENTARY NOTATION
Dissertation

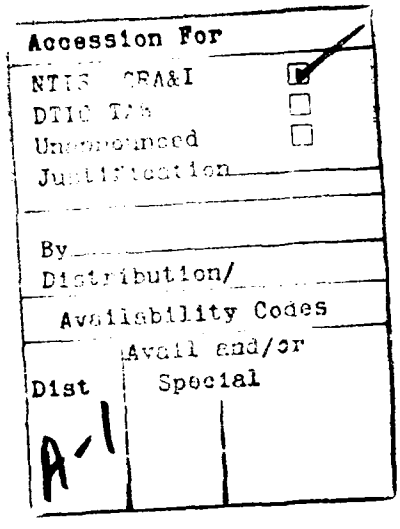
17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

BOZ is an automated graphic design and presentation system that designs graphics based on an analysis of the task for which a graphic is intended to support. When designing a graphic, BOZ aims to optimize two ways in which graphics help expedite human performance of information-processing tasks: (a) allowing users to substitute simple perceptual inferences in place of more demanding logical inferences; and (b) streamlining users' search for needed information. BOZ analyzes a logical description of a task to be performed by a human user and designs a provably equivalent perceptual task by substituting perceptual inference steps in place of logical inferences in the task description. BOZ then designs and renders an accompanying graphic, encoding and structuring date in the graphic such athat performance of each perceptual inference is supported and visual search is minimized. BOZ produces a graphic along with a perceptual procedure describing how to use the graphic to complete the task. A key feature of BOZ's task-

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION	
☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS		
22a NAME OF RESPONSIBLE INDIVIDUAL Dr. Alan L. Meyrowitz	22b TELEPHONE (Include Area Code) (202) 696-4302	22c OFFICE SYMBOL N00014





analytic approach is that it is able to design different presentations for the same information customized to the requirements of different tasks.

A second conponent of BOZ allows the logical and perceptual task descriptions to be directly used as cognitive simulations. The simulation component allows us to generate detailed theoretical predictions about the utility of any presentation with respect to a task. Simulaitons of logical and perceptual procedures track the number of inference steps and items searched for any combination of task procedure and presentation. Reaction time studies done with real users for one task show that a BOZ-designed graphic significantly reduces users' performance time to the task. Regression analyses link the observed efficiency savings to predictions about perceptual operator substitutions and pruning of visual search obtained using BOZ's simulation component. BOZ is used to design graphic presentations in a range of real-world task domains including customer airline reservations, class scheduling, a computer operator task, and statistical graphics.

TASK-ANALYTIC DESIGN OF GRAPHIC PRESENTATIONS

by

Stephen Michael Casner

B.S., Millersville University, 1984

M.S., University of Colorado, 1989

Submitted to the Graduate Faculty of

Arts and Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

1990

UNIVERSITY OF PITTSBURGH

FACULTY OF ARTS AND SCIENCES

TASK-ANALYTIC DESIGN OF GRAPHIC PRESENTATIONS

This dissertation was presented

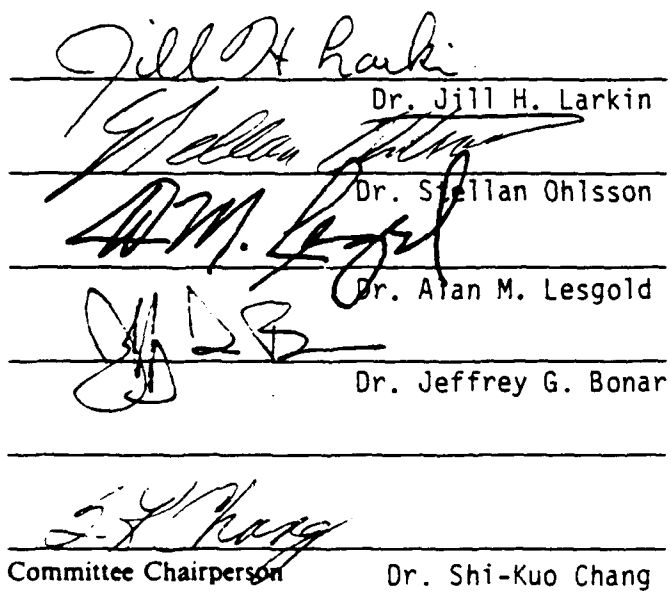
by

Stephen Michael Casner

It was defended on

May 18, 1990

and approved by



Dr. Jill H. Larkin

Dr. Stellan Ohlsson

Dr. Alan M. Lesgold

Dr. Jeffrey G. Bonar

Committee Chairperson Dr. Shi-Kuo Chang

© Copyright 1990

Stephen Michael Casner

TASK-ANALYTIC DESIGN OF GRAPHIC PRESENTATIONS

Stephen Michael Casner, Ph.D.

University of Pittsburgh, 1990

BOZ is an automated graphic design and presentation system that designs graphics based on an analysis of the task for which a graphic is intended to support. When designing a graphic, BOZ aims to optimize two ways in which graphics help expedite human performance of information-processing tasks: (a) allowing users to substitute simple perceptual inferences in place of more demanding logical inferences; and (b) streamlining users' search for needed information. BOZ analyzes a logical description of a task to be performed by a human user and designs a provably equivalent perceptual task by substituting perceptual inference steps in place of logical inferences in the task description. BOZ then designs and renders an accompanying graphic, encoding and structuring data in the graphic such that performance of each perceptual inference is supported and visual search is minimized. BOZ produces a graphic along with a perceptual procedure describing how to use the graphic to complete the task. A key feature of BOZ's task-analytic approach is that it is able to design different presentations of the same information customized to the requirements of different tasks.

A second component of BOZ allows the logical and perceptual task descriptions to be directly used as cognitive simulations. The simulation component allows us to generate detailed theoretical predictions about the utility of any presentation with respect to a task. Simulations of logical and perceptual procedures track the number of inference steps and items searched for any combination of task procedure and presentation. Reaction time studies done with real users for one task show that a BOZ-designed graphic significantly

reduces users' performance time to the task. Regression analyses link the observed efficiency savings to predictions about perceptual operator substitutions and pruning of visual search obtained using BOZ's simulation component. BOZ is used to design graphic presentations in a range of real-world task domains including customer airline reservations, class scheduling, a computer operator task, and statistical graphics.

FOREWARD

This work was supported by the Office of Naval Research, University Research Initiative Contract Number N00014-86-K-0678. I thank Keith Agee, Adam Beguelin, Stephanie Behrend, Brigham Bell, Mitchell Blake, Jeffrey Bonar, Sergio Bravo, Carol Casner, Frederick E. Casner III, Lynn Casner, Shi-Kuo Chang, David Connor, Ron Davis, Amy Doria, Anne Faltine, Jesús Gonzáles Ruiz, Bonnie John, Steve Johnson, Blair King, Ken Koedinger, Mark Kuta, Jill Larkin, Alan Lesgold, Clayton Lewis, Kendall Loughney, Alison Maddox, Tom Marion, Heather McQuaid, Allen Newell, Stellan Ohlsson, William Oliver, Coco Ortíz, John Pfeil, Perry Riggs, Doug Roesch, Paul Ross, Paul Ryan, Lael Schooler, Elaine Siemon, Gale Sinatra, Janice Singer, Stuart Smith, Suzanne Sundquist, Rich Thomason, Jorge Urzúa, Lalo Urzúa, Pita Urzúa, Alejandro Valdés, Roger Webster, Craig Wiederhold, George Wilson, and Winston for their help in getting me from there to here.

E também um pouco duma raça,
Que não tem medo de fumaça,
Que não se entrega não.

TABLE OF CONTENTS

Page

FORWARD i

INTRODUCTION 1

1. THE COGNITIVE UTILITY OF GRAPHIC PRESENTATIONS 5
1.1. Computational Advantages 5
1.2. Search advantages 7
1.3. Summary 14

2. RELATED WORK 16
2.1 Early Graphic Design 16
2.2 Graphic Design Practices 16
2.3 Automated Graphic Presentation Tools 22
2.4 Experimental Studies of People Using Graphics 27
2.4.1 Studies of Human Visual Perception 27
2.4.2 Studies of Complex Task Performance Using Graphics 37
2.5 Information-Processing Models of Graphic Design and Use 42
2.6 Conclusions 44

3. TASK-ANALYTIC DESIGN OF GRAPHIC PRESENTATIONS 46
3.1 Logical Task Description Language 49
3.2 Perceptual Operator Substitution 54
3.2.1 A Catalog of Perceptual Operators 54
3.2.2 Substituting Operators 57
3.3 Perceptual Data Structuring 64
3.3.1 Operator Vectors 65
3.3.2 Relationships Between Vectors 66
3.4 Perceptual Operator Selection 68
3.4.1 Human Performance Rankings for Perceptual Operators 69
3.4.2 Primitive Graphical Language Expressiveness 71
3.4.3 Operator Combinability 71
3.5 Graphic Presentation Rendering 76
3.5.1 Translating Logical Facts to Structured Graphical Facts 76
3.5.2 Rendering Graphical Facts 77
3.5.3 Interactive Graphic Presentation Objects 80
3.6 Limitations of BOZ's Automated Design Approach 80
3.6.1 Limitations of the Task Description Language 80
3.6.2 Limitations of the Perceptual Operator Substitution Component 81
3.6.3 Limitations of the Automated Perceptual Operator Selection Component 82
3.6.4 Limitations of the Automated Rendering Component 85

4. THEORETICAL MEASURES OF GRAPHIC DESIGN EFFECTIVENESS 88
4.1 Perceptual Computation 88

4.1.1 Substituting Perceptual Operators 89
4.1.2 Operator Elimination 90
4.1.3 Hypothesized Computational Advantages of the Airline Graphic 92
4.2 Visual Search 93
4.2.1. Locality 93
4.2.2 Indexing 95
4.2.3 Parallelized Operators 96
4.2.4 Predicted Search Advantages of the Airline Graphic 98

5. GRAPHIC DESIGN EXAMPLES 102
5.1 An Extended Set of Airline Reservation Tasks 102
5.2. SPOOL File Management 120
5.3. Class Scheduler Interface 127
5.4. Information Graphics 132
5.4.1 Consumer Report 132
5.4.2. Employees 135
5.5. Conclusion 138

6. EXPERIMENTAL STUDY OF GRAPHIC DESIGN EFFECTIVENESS 139
6.1 Method 139
6.2 Theoretical Predictions 142
6.3 Results and Discussion 144
6.4 Conclusion 148

7. CONCLUSIONS AND DISCUSSION 149
7.1 Summary 149
7.2 Contributions 149
7.3 Limitations of BOZ 152
7.3.1 Cognitive Issues 152
7.3.2 Automated Graphic Design Issues 158
7.4 Other Advantages of BOZ-Designed Graphic Presentations 160
7.5 Advantages of Graphic Presentations Not Addressed by BOZ 161
7.5 Further Issues 163
7.6 Concluding Remarks 164

APPENDIX 167

BIBLIOGRAPHY 176

LIST OF TABLES

Table 1: Six Task-Specific Advantages of Graphic Presentation Use
Table 2: The Primitive Graphical Languages
Table 3: Perceptual Operators
Table 4: Equivalence Classes for Perceptual Operators
Table 5: Members of Two Operator Equivalence Classes
Table 6: Ranking of Perceptual Operators and Equivalence Classes
Table 7: Graphic Presentation Objects of the Primitive Graphical Languages
Table 8: Composition Rules for Graphic Presentation Objects
Table 9: Predicted Computational Advantages of the Airline Schedule Graphic
Table 10: Predicted Search Advantages of the Airline Schedule Graphic
Table 11: Predicted Efficiency Advantages of the Specific Layover Graphic
Table 12: Predicted Efficiency Advantages of the Cheapest Flight Graphic
Table 13: Predicted Efficiency Advantages of the Minimum Connections Graphic
Table 14: Predicted Efficiency Advantages of the SPOOL Graphic
Table 15: Predicted Efficiency Advantages of the Class Scheduling Graphic
Table 16: Predicted Computational Advantages of the Four Experimental Airline Schedules
Table 17: Predicted Search Advantages of the Four Experimental Airline Schedules

LIST OF FIGURES

Figure 1: A Graphic Presentation That Supports Efficient Perceptual Operators.
Figure 2: A Graphic Presentation That Features An Emergent Property.
Figure 3: A Presentation That Uses Multi-Dimensional Graphical Objects.
Figure 4: A Graphic Presentation That Groups Related Quantities in a Single Locality.
Figure 5: A Tabular Presentation That Supports Spatial Indexing.
Figure 6: A Graphic Presentation That Supports Retinal Indexing.
Figure 7: A Tabular Presentation That Supports Retinal Indexing.
Figure 8: A (Very) Crooked Bar Chart
Figure 9: A Second Type of Graphical Lie.
Figure 10: Experimental Studies of Complex Task Performance.
Figure 11: Overview of BOZ
Figure 12: Logical Airline Reservation Procedure
Figure 13: Factbase for the Airline Reservation Task
Figure 14: Operator Classifications for the Airline Reservation Task
Figure 15: Feature Space for the Airline Reservation Task
Figure 16: Vectors for the Airline Reservation Task.
Figure 17: Vector Relationships for the Airline Reservation Task
Figure 18: Initial Perceptual Data Structure Specification for the Airline Reservation Task
Figure 19: The Perceptual Airline Reservation Procedure
Figure 20: Final Perceptual Data Structure Specification for the Airline Reservation Task
Figure 21: Example Graphical Facts
Figure 22: Translated Airline Reservation Facts
Figure 23: Structured Graphical Facts
Figure 24: Rendered Graphic Airline Schedule.
Figure 25: Rendered Seating Chart.
Figure 26: Roth's Relational Data Types
Figure 27: Pictograms: A Type of Graphic Not Designable by BOZ
Figure 28: Operator Elimination in the Perceptual Airline Reservation Procedure
Figure 29: A Set of Grouped Facts
Figure 30: A Series of Lookup Operators

Figure 31: The rightOfSearchU Parallelized Procedure
Figure 32: Tabular Airline Schedule Presentation.
Figure 33: Simulation Results for the Alternative Procedures and Presentations.
Figure 34: Logical Operators for Airline Reservation Task 1.
Figure 35: Logical Procedure for Airline Reservation Task 1.
Figure 36: Graphic Presentation Designed for Airline Reservation Task 1.
Figure 37: Perceptual Procedure for Airline Reservation Task 1.
Figure 38: Simulation Results for Airline Reservation Task 1.
Figure 39: Logical Operators for Airline Reservation Task 2.
Figure 40: Logical Procedure for Airline Reservation Task 2.
Figure 41: Graphic Presentation Designed for Airline Reservation Task 2.
Figure 42: Perceptual Procedure for Airline Reservation Task 2.
Figure 43: Simulation Results for Airline Reservation Task 2.
Figure 44: Alternative Graphic Presentation for Airline Reservation Task 2.
Figure 45: Logical Operators for Airline Reservation Task 3.
Figure 46: Logical Procedure for Airline Reservation Task 3.
Figure 47: Graphic Presentation Designed for Airline Reservation Task 3.
Figure 48: Perceptual Procedure for Airline Reservation Task 3.
Figure 49: Simulation Results for Airline Reservation Task 3.
Figure 50: Logical Operators for the SPOOL Task.
Figure 51: Logical Procedure for the SPOOL Task.
Figure 52: Graphic Presentation Designed for the SPOOL Task.
Figure 53: Simulation Results for the SPOOL Task.
Figure 54: Logical Operators for the Class Scheduling Task.
Figure 55: Logical Procedure for the Class Scheduling Task.
Figure 56: Graphic Presentation Designed for the Class Scheduling Task.
Figure 57: Perceptual Procedure for the Class Scheduling Task.
Figure 58: Simulation Results for the Class Scheduling Task.
Figure 59: Logical Operators for the Consumers Task.
Figure 60: Graphic Presentation Designed for the Consumers Task.
Figure 61: A Second Graphic Presentation for the Consumers Task.
Figure 62: Logical Operators for Employees Task 1.
Figure 63: Graphic Presentation Designed for Employees Task 2.
Figure 64: Logical Operators for Employees Task 2.
Figure 65: Graphic Presentation Designed for Employees Task 2.
Figure 66: Graphic Presentation Designed for A Combination of Employees Tasks 1 & 2.
Figure 67: Four Experimental Graphics.

Figure 68: Participants' Mean Performance Times for the Airline Reservation Task.
Figure 69: A Database Query Task Description Language.
Figure 70: A Graphic Presentation That Provides Control Information.
Figure 71: A Memorable Graphic Presentation.

INTRODUCTION

Our intuitions about the utility of graphic presentations of information have long suggested that graphics are indeed a useful tool for communicating, comprehending, storing, and analyzing information. Many striking examples of graphics seem to "reveal" characteristics of a data set or help us reason more efficiently about information that expressed in a different format would seem to require more work or be prohibitively complex. Consequently, it is no surprise that graphics enjoy widespread use. While efforts to create interesting graphic presentations date back to the late 1700's, it has been only recently that investigators have concerned themselves with empirically testing the utility of graphics as artifacts to support information-processing tasks. A striking conclusion of recent empirical studies[1] is that it is a false assumption that graphic presentations are inherently better than other presentations, or that perceptual inferences are always made more efficiently than non-perceptual inferences. Rather, these studies suggest that the usefulness of a graphic is a function of the *task* that the graphic is being used to support. Twenty-nine independent empirical studies surveyed in Jarvenpaa and Dickson (1988) found graphics superior to tabular presentations for a restricted set of information-processing tasks, and observed no benefits or poorer performance for other tasks. Examples of graphics that succeed in practice are best explained as "situationally dependent artifacts" whose success arises out of the combination of task performed and the particular graphic used. Generalizations made about the observed usefulness of a graphic for one task are highly inappropriate since using the same graphic for different tasks typically cause the usefulness of the graphic to

[1] Experimental studies of the practical utility of graphic presentations have been conducted mainly in the fields of Management Information Systems [Benbasat et al, 1986; Dickson et al, 1986; Jarvenpaa and Dickson, 1988] and Human Factors [Salomon, 1972; Gibson and Laios, 1978; Krohn, 1983; Robinson and Eberts, 1987; Tullis, 1981].

disappear. Consequently, graphic design principles that focus on an analysis of the information to be presented in a graphic, such as "line graphs are best for continuous data," are too under-specified to be useful in general. That is, empirical studies have shown that line graphs are supportive of some tasks that manipulate continuous data and are detrimental to the performance of others. The implication is that effective graphic design should begin with the task that a graphic is intended to support, and be focused on finding those parts of a task, if any, that might be performed more efficiently within the context of a graphic presentation. Analysis of the information to be presented in a graphic is assigned a secondary role.

Adopting the view that there exists a strong tie between the utility of a graphic presentation and the purpose for which it is intended defines three important sub-goals in developing a scientific treatment of the design of graphic presentations to support information-processing tasks. First, there is a long-standing challenge to replace intuitive notions with a more detailed and scientific understanding of how graphics and the procedures used to interact with them leverage problem solving and task performance. Second, there is a need to articulate a *prescriptive* task-based design theory for graphic presentations. Such a theory would begin with an understanding of a task to be supported and guide the designer in constructing a graphic, applying specific principles of how graphics support tasks at each step, placing analysis of the information to be presented in a secondary role. Third, there is a need to establish techniques for predicting the utility of a graphic presentation with respect to a task. These techniques would measure important parameters when combinations of presentations and tasks are considered. These parameters would enable theoretical predictions about the utility of a presentation purported to support an information-processing task, as well as meaningful comparisons between alternative presentations.

Larkin and Simon (1987) in an earlier treatment of graphics, aimed to articulate a set of scientific principles that describe why performing an information-processing task using a

graphic under some circumstances provides advantages to the user. An important premise of Larkin and Simon's work is that, when comparing alternative presentations, it is fruitful to characterize graphic-based problem solving using the same information-processing models used to help understand problem solving using other representations [Newell and Simon, 1972]. Characterizing graphics and tasks using familiar information-processing models allowed Larkin and Simon to make detailed comparisons between "logical" tasks, i.e., tasks performed outside the context of any particular representation, and perceptual tasks performed using a graphic. Larkin and Simon's analysis showed that comparisons between tasks performed using alternative informationally equivalent representations could help lead to the discovery of concrete ways in which graphics support task performance. Larkin and Simon's work produced two general principles of how graphics support tasks. Graphics often allow users to: (1) substitute quick perceptual judgements in place of more demanding logical inferences; and (2) to expedite search for needed information.

This dissertation explores an approach to the design of graphic presentations based on an analysis of the tasks for which they are intended to support. The design approach is implemented in an automated graphic design and presentation tool called BOZ. The core idea behind BOZ can be summarized as follows. *Since the potential advantages of graphics are task-related, graphic design activities should focus on designing efficient perceptual tasks. Decisions made about how to encode and structure information in an accompanying graphic should be based primarily on supporting efficient and accurate performance of the perceptual task.* The enabling step in the task-analytic approach is to capture the notion of a perceptual task performed by human users using the same formal framework used to describe other types of information-processing tasks, allowing design decisions to follow formal criteria.

Chapter 1, drawing on Larkin and Simon's work, develops a theoretical analysis of the computational and search advantages that arise during execution of graphic presentation-

based problem-solving procedures. Chapter 2 reviews other work related to the problem of designing graphic presentations, dating from William Playfair's efforts in the late 1800's up to recent cognitive and experimental research that has directly influenced the ideas presented in this dissertation. Chapter 3 describes an automated graphic design and presentation system called BOZ that approaches the design problem from a task perspective. BOZ analyzes a procedural description of a user task and derives a provably equivalent perceptual procedure by substituting perceptual inference steps in place of logical inferences. BOZ automatically designs and renders an accompanying graphic, encoding data in the graphic such that performance of each perceptual inference is supported and visual search is minimized. Chapter 4 treats in more depth the computational and search advantages of perceptual procedures and shows how quantitative predictions can be made about differences in computational and search efficiency between alternative presentations. A simulation tool is described that automates the process of generating theoretical predictions about the utility of a presentation with respect to a task. The simulation component automatically creates executable simulations from the procedural task descriptions submitted to BOZ as input, and from the perceptual procedures produced by BOZ. These simulations are used to make quantitative comparisons between alternative presentations and procedures. Chapter 5 develops several examples of how BOZ is used to design graphic presentations to support practical real-world tasks. Chapter 6 describes an experiment in which participants were asked to use a set of alternative presentations to compare participants' performance with theoretical predictions obtained using simulations. Chapter 7 discusses the implications and limitations of BOZ as a theory of graphic presentation design, an automated graphic design tool, and as a computational model of human problem solving within the context of graphic presentations.

CHAPTER 1

THE COGNITIVE UTILITY OF GRAPHIC PRESENTATIONS

Larkin and Simon (1987) made two general points about the utility of graphic presentations: (1) that graphic presentations could sometimes help reduce the amount of mental computation required to complete a task; and (2) that using a graphics sometimes allows users to spend less time searching for needed information. This chapter develops a more detailed breakdown of these two types of advantages and relates them to specific graphical techniques available for use when designing presentations.

1.1. Computational Advantages

Larkin and Simon's first point about the utility of graphic presentations was that graphically expressed information sometimes allows users to do less mental computation than is necessary outside the context of a graphic, or to do computational steps that are performed more efficiently. This analysis is expanded in the present work to explore three specific types of computational advantages offered by graphics.

Substituting operators. Graphics often allow users to substitute less demanding perceptual operators in place of more complex logical operators. Perceptual operators (e.g., distance and color comparisons, spatial coincidence judgements) can often give users the same information as more complex logical operators. The scatter plot shown in Figure 1 is an example that illustrates how human performance of a set of perceptual operators can be observably more efficient than performance of informationally equivalent logical operators. When interpolating values between points using the scatter plot, users can substitute the task of perceptually constructing a line connecting two adjacent points and

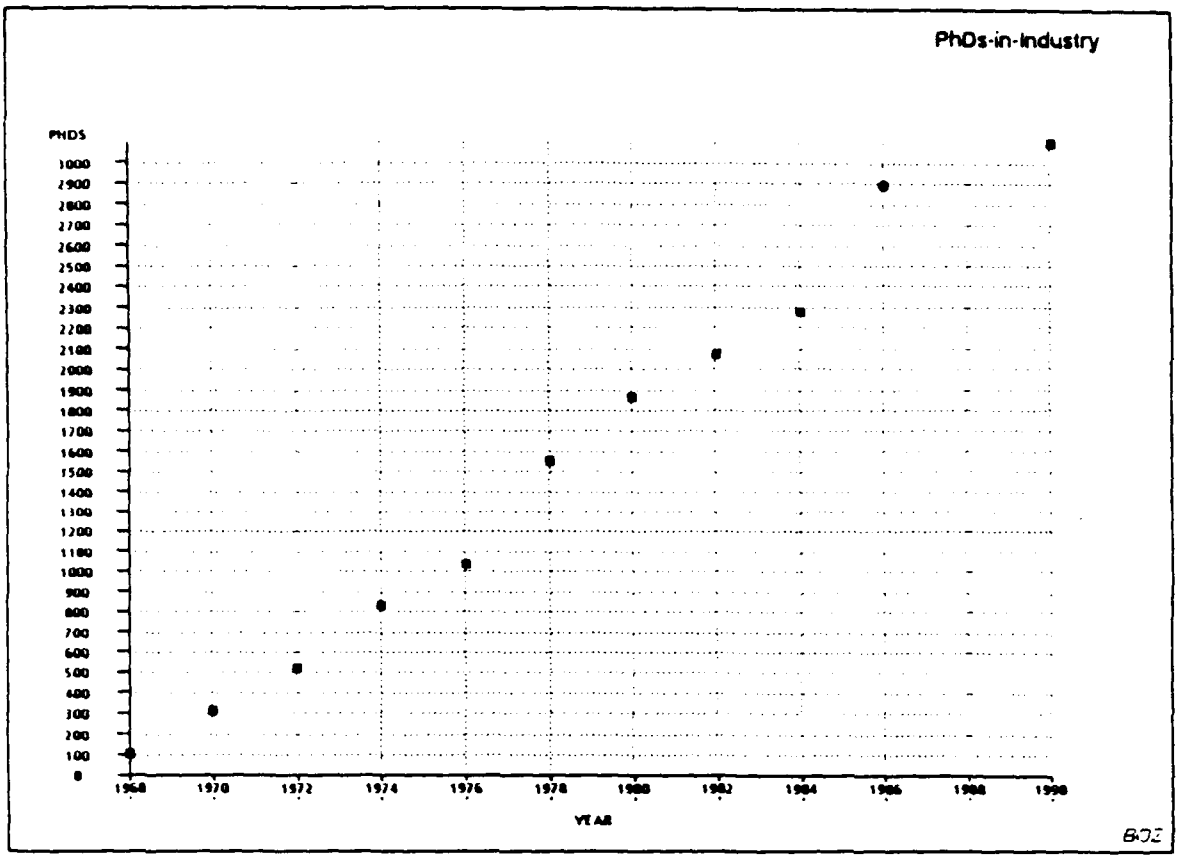


Figure 1: A Graphic Presentation That Supports Efficient Perceptual Operators.

looking up values along that line, for the task of subtracting and dividing two data values. For large data sets, repeatedly performing this efficient perceptual task allows users to quickly understand overall trends in the data.

Step skipping. Graphics sometimes allows users to omit steps that are otherwise necessary when a task is performed without a graphic. For example, when determining absolute differences between two biannual figures using the scatter plot in Figure 1, there is no need to determine the values for the two years. That is, the user can simply determine the vertical distance between two points in the plot and report the answer. The steps that determine the data values themselves can be skipped. Producing the same answer using

numerically expressed information would require that the data values for the two individual years be known and then subtracted.

Emergence. Graphic presentations sometimes encode information additional to what is intended. A graphic devised by Ohlsson (1987) for teaching fractions concepts (shown in Figure 2) demonstrates a property called *emergence*.

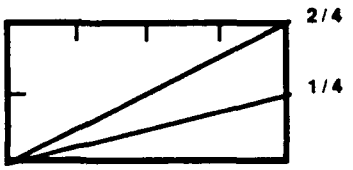


Figure 2: A Graphic Presentation That Features An Emergent Property.

Ohlsson's analysis of the Rectangles World shows that the numerator and denominator of a fraction are represented by the height and width of a rectangle, respectively. However, when these two encodings are used and the diagonal of the rectangle is drawn, the slope of the diagonal encodes the value of the fraction. Consequently, comparing two fractions is now reduced to the task of comparing the slopes of two lines. There is no need to compute the value of the fractions since they are explicitly represented through an emergent property of the graphic presentation (slope). Mackinlay and Genesereth (1985) generalize this property to show that it is an unavoidable phenomenon to express facts using some graphical dimensions without expressing facts in other dimensions.

1.2. Search advantages

Larkin and Simon's second basic point about the utility of graphic presentations was that they sometimes allows users to reduce the time spent searching for information. This analysis is developed in the present work to illustrate three ways in which graphics help users reduce search for information.

Locality. Objects in a graphic can have many perceptual properties including spatial position, size, shape, color, labels, etc. Graphics help users save time when searching for needed information when they group several related dimensions of information in a single graphical object by encoding that information using various perceptual dimensions of the graphical object. The graphic in Figure 3 demonstrates this technique.

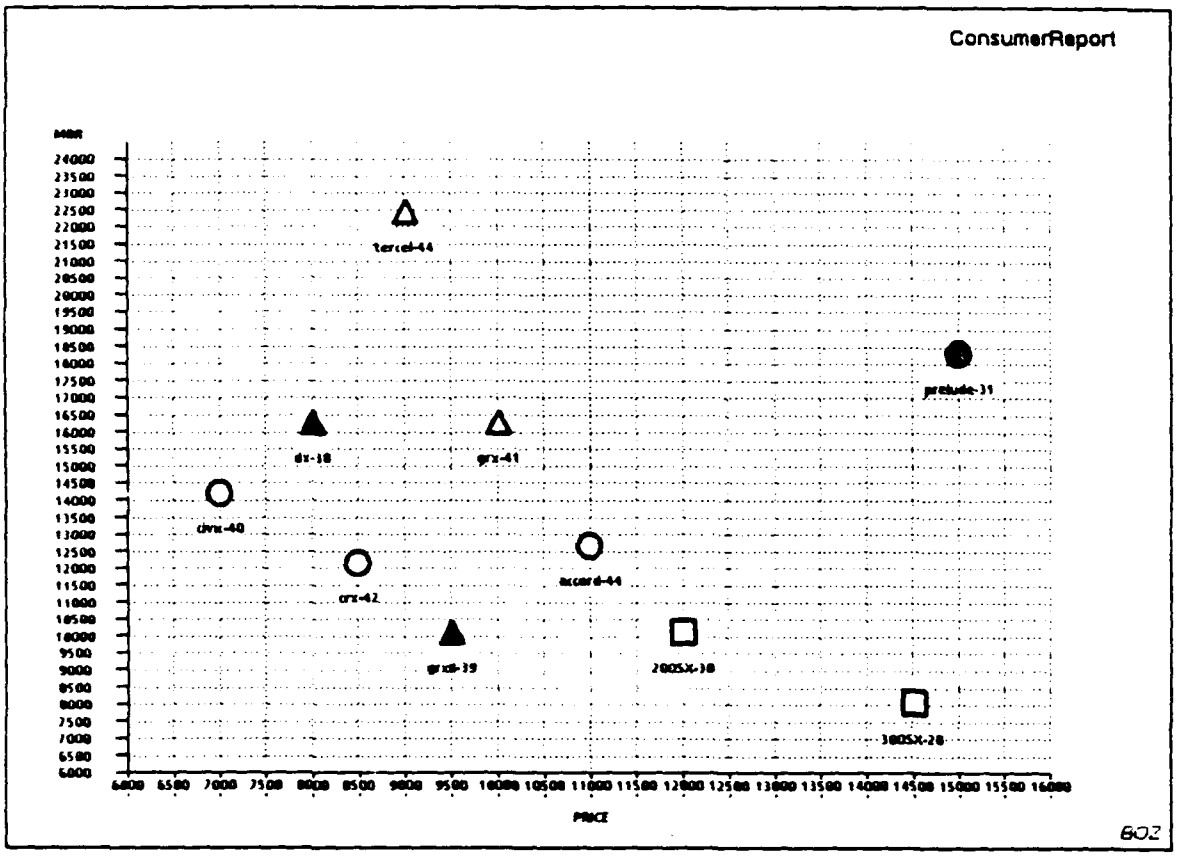


Figure 3: A Presentation That Uses Multi-Dimensional Graphical Objects.

The multi-dimensional graphical objects in Figure 3 reduce the need for the eye to travel between items in the graphic. Each object in the graphic in Figure 3 encodes six dimensions of information about cars: car manufacturers (shape), the models they produce (labels), price (horizontal position), miles per gallon (labels), type of safety features (shading), and miles between repairs (vertical position). When looking up relevant

information about a single car search time is reduced because no eye movement is necessary between items.

Vector diagrams, shown in Figure 4, are a second example of a graphic presentation that encodes multiple dimensions of information in the same locality. Each vector in the plane encodes information about direction (slope), magnitude (length), and whether or not the vector has been derived from other component vectors (thickness).

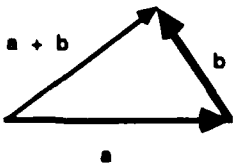


Figure 4: A Graphic Presentation That Groups Related Quantities In A Single Locality.

Search reductions due to locality are constrained by limitations on short-term memory in two important ways. First, as the number of perceptual dimensions used in a single graphical object increases, it becomes increasingly difficult to maintain all properties of that object in memory at one time [Ericsson et al, 1980]. If the number of dimensions used in any single perceptual inference exceeds four, it is likely that the user will need to perceive the object several times. Second, as the number of different values that each perceptual dimension can take on increases, it may be necessary for the user to consult a key or legend where this information is permanently recorded, again causing search reductions to disappear.

Indexing. Several perceptual dimensions used to encode information in graphic presentations, such as spatial position, color, shading, texture, and shape, can be processed by humans pre-attentively [Treisman, 1977; Treisman and Gelade, 1980; Beck and Ambler, 1973; Pomerantz et al, 1977; Julesz, 1981]. Information encoded in a graphic using these perceptual dimensions can be perceived and understood without requiring the

eye to be fixed at the location where that information is expressed. Perceptual dimensions that do not support pre-attentive processing require eye fixations for all items considered during search [Nickerson, 1966]. Perceptual dimensions that support pre-attentive processing are often referred to as *indexing* dimensions [Ullman, 1984] since they allow users of a graphic to quickly partition or index the items appearing in a graphic along a particular dimension of information. Two types of indexing perceptual dimensions are distinguished here.

Spatial indexing perceptual dimensions allow users of a graphic presentation to single out a subset of items by following a restricted path of eye movement over the graphic. The tabular presentation in Figure 5 uses spatial indexing to partition the data pertaining to tourism in South American countries among the rows and columns of a table.

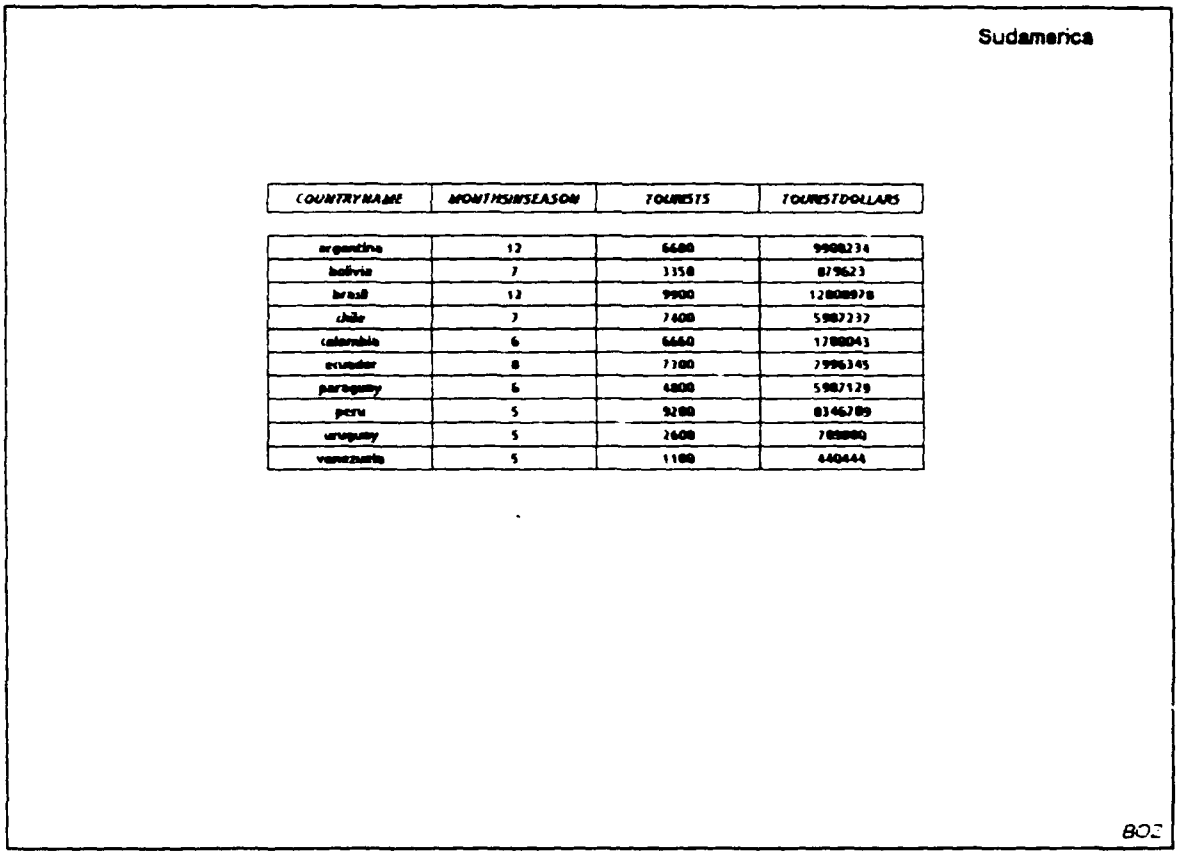


Figure 5: A Tabular Presentation That Supports Spatial Indexing.

To locate all information pertaining to a particular country, a user can simply scan the relevant row which contains this information. Similarly, using the graphic presentation in Figure 3, when looking for a low-priced car users can limit their search to cars appearing along the bottom of the graphic since the vertical position of each car is being used to denote price.

Retinal indexing allows users to immediately attend to items having particular data values when those values are encoded using one of the retinal indexing perceptual dimensions such as color, texture, or shape. For example, the plot chart in Figure 6 uses squares positioned in the plane to represent sales totals and number of years worked by sales managers within a company, and shading to encode their retirement status.

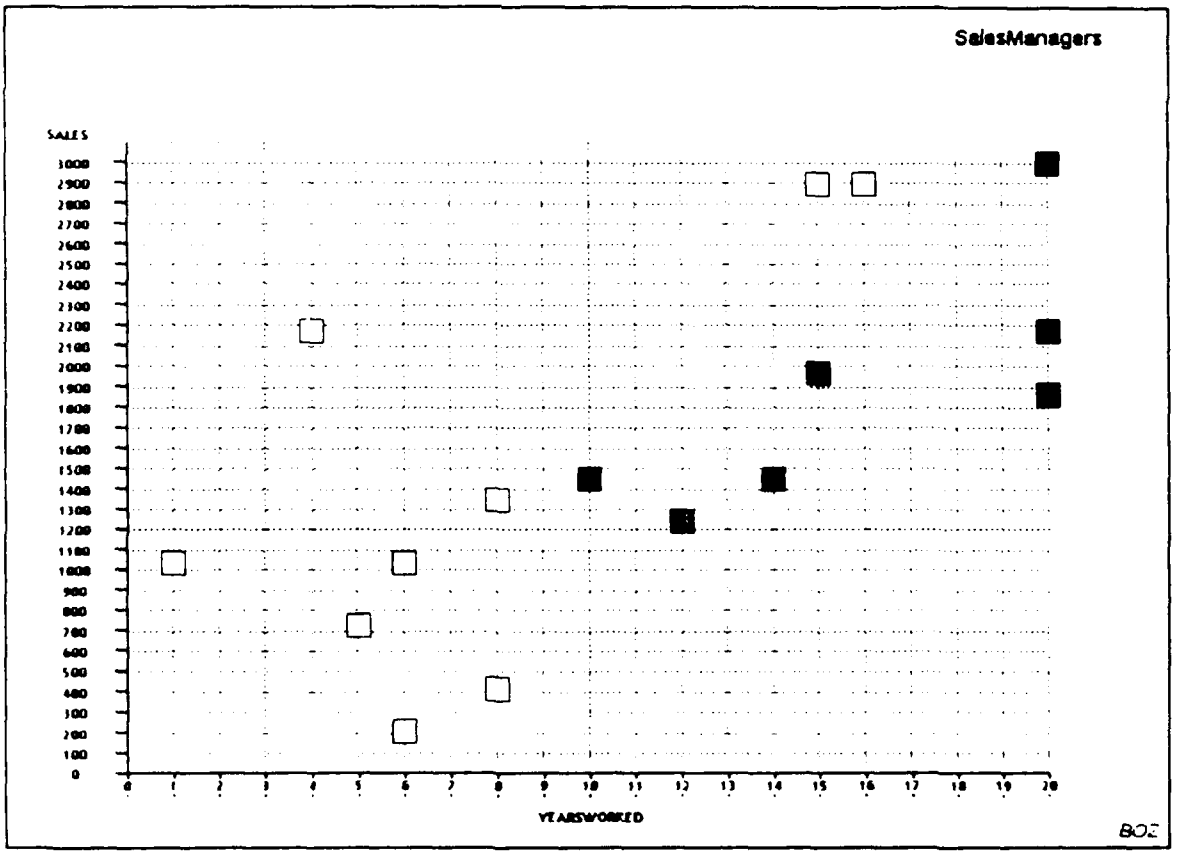


Figure 6: A Graphic Presentation That Supports Retinal Indexing.

Using the graphic in Figure 6, when looking for a retired sale manager, the user can immediately rule out from their search all unshaded squares, reducing the total items searched to seven. Green and Anderson (1956) provide evidence that users are indeed able to immediately rule out all items not having the target color, limiting visual search to qualifying items only. Investigators have shown that several other perceptual dimensions can sometimes be processed pre-attentively when they are decoded using the same strategy used to decode some other pre-attentive perceptual dimension. Neisser (1964) showed that words can sometimes be identified in a list by attending only to their shape or size when the number of unique words is small. For example, when using the tabular sales managers presentation in Figure 7, users can achieve the same search savings obtained using the graphic presentation when searching for a retired manager.

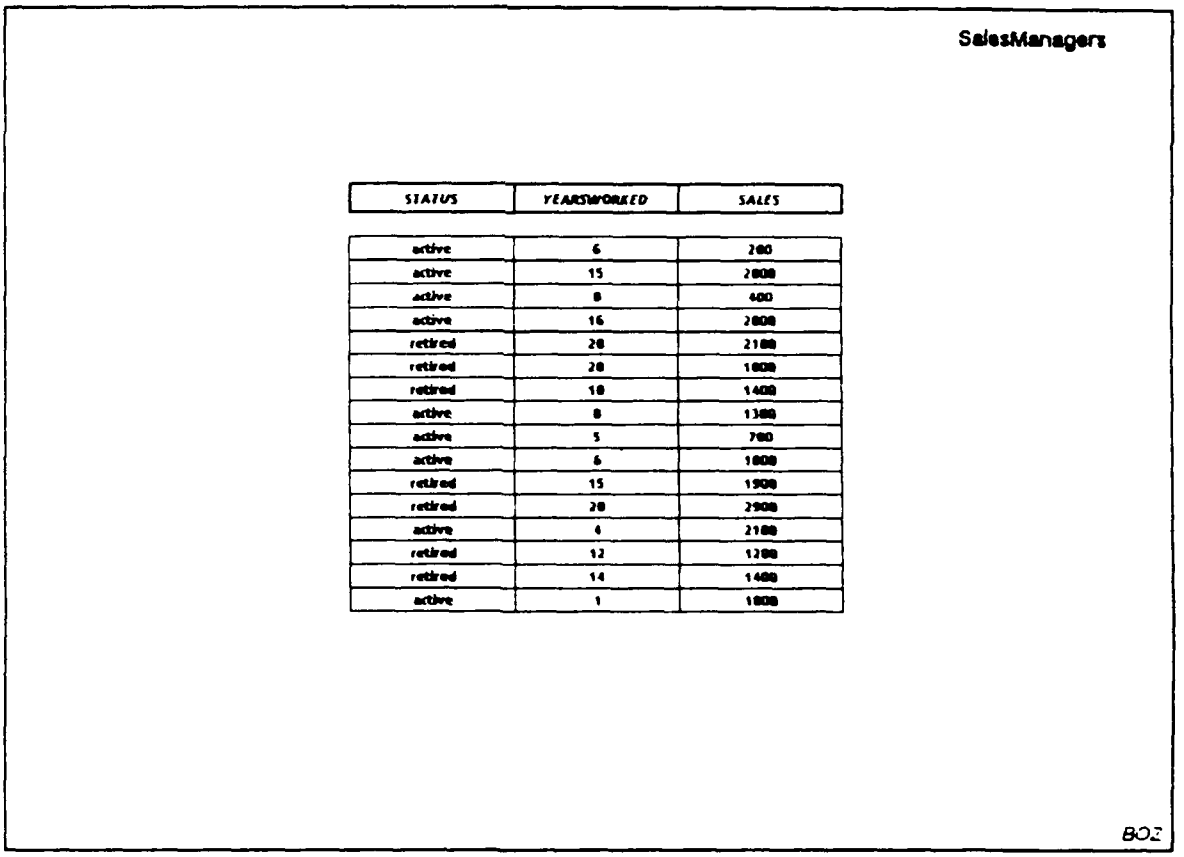


Figure 7: A Tabular Presentation That Supports Retinal Indexing.

Since only two words are needed to encode the different retirement statuses (i.e., active and retired), Neisser's data suggests that users are able to pre-lexically recognize the word "retired" in the third column by considering only its shape.

Parallelized operators. Earlier studies suggest that the efficiency at which perceptual operators are performed, or the nature of the brain hardware that implement them, allows some perceptual operators to be executed in parallel. This capability allows the user to expedite the process of computing visual search constraints and to rule out many items in a graphic presentation that would otherwise need to be considered. While the question of whether or not perceptual operators can truly be executed in parallel remains open for debate, experimental evidence suggests that it is indeed the case that several operators can be performed in less than the sum of the times that would be required to perform each operator individually.

Ullman (1984) distinguishes two[1] types of parallelism. *Spatial parallelism* occurs when a single perceptual operator is simultaneously executed at more than one spatial location within a graphic. For example, the task of comparing the colors of two car symbols in Figure 3 is an example of spatial parallelism when the task is performed by fixing the eye at a position between the two symbols. There are two important limitations on the examples of spatial parallelism we can hypothesize. First, spatial parallelism relies on the phenomenon referred to as retinal indexing above. Since the eye cannot be fixed at more than one spatial location at a time, the operator that we claim to be performed simultaneously at separate locations cannot require eye fixation. Second, spatial parallelism can succeed only when the complete set of spatial locations where the operator is to be applied occur within a range of roughly 3° from the point of eye fixation.

[1] Ullman (1984) describes a third kind of parallelism called *temporal parallelism*. Temporal parallelism applies a sequence of perceptual operators, pop_1, pop_2, ..., pop_n, to the items in a temporal string input, I_1, I_2, ..., I_m. The application of the operators is parallelized in the following way. pop_1 is first applied to I_1. Next, as pop_2 is being applied to I_2, pop_1 is simultaneously applied to I_2.

Functional parallelism occurs when two or more unique perceptual operators are executed simultaneously at the same spatial location within a graphic. For example, when the eye is fixed on a particular vehicle in the presentation in Figure 3, users can simultaneously determine the make (shading) and model (shape) of that vehicle.

It is important to note that functional parallelism cannot be performed simultaneously with spatial parallelism. Treisman (1977) and Treisman and Gelade (1980) asked participants to locate a unique green T within a presentation containing brown Ts and green Xs. This task allows the possibility to exploit both types of parallelism simultaneously. That is, the fastest strategy is one in which the perceptual operators of judging the shape of a letter and determining the color of a letter are performed simultaneously (functional parallelism) at multiple locations (spatial parallelism).

1.3. Summary

The above discussion describes six observations that comprise what we currently understand to be the ways in which the use of a graphic presentation can expedite the performance of an information-processing task. Table 1 summarizes these observations. It was shown that combinations of tasks and presentations can be found in which the advantages described in Table 1 arise. The purpose of this dissertation is to address the more general research question:

> *Is there a principled theory of graphic presentation design that proceeds by analyzing the specifics of a task to be supported and that produces a graphic presentation that best exploits each of the six advantages that graphic presentations can offer?*

Table 1: Six Task-Specific Advantages of Graphic Presentation Use

1. **Substituting Operators**: Efficiently performed perceptual operators sometimes give users the same information as more demanding logical operators.

2. **Step Skipping**: Perceptual operators that compute or compare values do not require the user to explicitly know the values of the data upon which they operate.

3. **Emergence**: Some graphical conventions encode information additional to what is explicitly intended, allowing the user to obtain some information more easily.

4. **Locality**: Graphical objects in a presentation can exploit the many perceptual dimensions to encode several dimensions of information at a single spatial location, eliminating search when collecting data values pertaining to a single object or entity.

5. **Indexing**: Some perceptual dimensions can be perceived pre-attentively. This feature allows the user to immediately attend to graphical objects having "odd man out" graphical properties avoiding the need to consider all graphical objects in a presentation during search.

6. **Parallel Operator Performance**: Some perceptual inferences can be performed in parallel, allowing user to apply multi-dimensional search criteria during visual search. The space of graphical objects considered during search is reduced by simultaneously disqualifying all graphical objects not implicated by the search criteria.

CHAPTER 2

RELATED WORK

2.1 Early Graphic Design

William Playfair (1759-1823) and Johann Heinrich Lambert (1728-1777) are generally credited with introducing modern information graphic design.[1] Playfair, a British political economist, in a series of three books [Playfair, 1786, 1798, 1801] set forth a family of novel graphic techniques for displaying statistical data.[2] Playfair pointed out that the usefulness of his inventions rested in the observation that "as much information may be obtained in five minutes as would require whole days to imprint on the memory, in a lasting manner, by a table of figures [Playfair, 1786]." Many of Playfair's graphical inventions, such as the bar chart, pie chart, circle graph, and rectilinear coordinate graph, comprise many of the popular graphic designs used today. J. H. Lambert, a Swiss-German scientist, is credited with adopting the time-series graph (the most popular form of graphic used today) into scientific use.[3]

2.2 Graphic Design Practices

Scores of books on the topic of graphic presentation appeared immediately after Playfair's early works and continue to be written today. Early books such as Karsten (1923), Haskell (1926), Modley (1932), Brinton (1939), and Hall (1943) aimed to provide the

[1] Arnheim (1969) makes the point that one reason for the delayed introduction of graphical presentations for scientific purposes is that perception was traditionally considered to be inferior to other thought processes since is did not involve the use of language.

[2] Playfair's inventions, of course, draw on many ingenious inventions that pre-dated him, such as the Cartesian coordinate system, analytic geometry, and cartographic maps. Funkhouser (1937) provides an interesting discussion of the origins of graphical displays.

[3] Interestingly, the invention of the time-series graph may pre-date Lambert by as much as 800 years! Funkhouser (1936) presents an example of a time-series graph depicting planetary orbits believed to have been created sometime during the tenth or eleven century (cited in Tufte (1983)).

reader with a collection of formats popularly used for graphically presenting data. These books focused on explaining the graphical conventions used to portray data and the situations in which each different graphic technique was believed to be most effective. Explanations of why and when different graphic designs were useful relied mainly on informal analyses of the information to be portrayed in the graphic.

Lockwood (1969), Tufte (1983), Tufte (1990), Martin and McClure (1985), McCleary (1981), Cleveland (1985), and Schmid (1983) comprise a second generation of information graphic design books. These books extend earlier works in an important way by advancing more sophisticated design principles that are sensitive to many observed features of human visual perception such as perceptual distortion, salience, and the negative effects of graphical decorations that do not encode relevant information (i.e., "chart junk [Tufte, 1983]"), and provide useful advice about how to insure correct interpretation of graphical conventions. Kosslyn (1985) reviews several of these books and the perception-related issues they address. An important limitation of the books is that, even though several of the books argue that the purpose of a graphic is a first design concern [Schmid, 1983], they provide limited prescriptive information about how to match graphics to user tasks. Some design principles focus entirely on an analysis of the information to be presented. For example, Tufte (1983) argues that "time-series presentations are at their best for big data sets with real variability." These principles assume that the user will use the graphic for a single task: to globally comprehend and summarize all of the information presented in the graphic. Tufte illustrates the usefulness of the time-series presentation with a famous graphic presentation of train schedule information developed in the late 1800's [Marey, 1885]. It is not clear that anyone (other than railroad company management) would ever use the presentation to comprehend or summarize all of the information in the train schedule graphic. A more likely task would be to find a single train that meets a particular user's time and place constraints. Other design principles do include task-specific criteria. Schmid (1983) points out that "the bar chart is characteristically used for direct

comparisons of magnitude for descriptively labelled categories." Schmid's task-based principles, however, are limited in two important ways. First, the tasks are usually limited to simple data look-up and comparison tasks. This overlooks many of the uses of graphics for supporting more sophisticated computational tasks such as arithmetic, proportional reasoning, and interpolation, and does not recognize the usefulness of graphics for supporting visual search tasks. Second, the principles do not provide information about how to combine the features of two or more graphic designs to arrive at original designs to support novel tasks.

Jacques Bertin's *Sémiologie Graphique*, first published in 1967, deserves special mention as due to its depth of analysis unmatched by any other book on the subject of information graphic design. Bertin's design approach is based on the premise that the representational powers of "visual variables," or perceptual dimensions used to encode information in a graphic, can be formally articulated. The design of a graphic presentation proceeds by analyzing the type and size of each domain set of information to be depicted in a graphic and matching each domain set with that visual variable whose representational power best fit the requirements of the domain set.[4] Bertin's work recognizes the importance of the tasks to be performed with a graphic but only treats a limited set of tasks for which graphics are useful: value look up, recognizing global trends, and improving memory for data. Other more complex tasks involving perceptual computation and visual search are not considered and do not follow immediately from Bertin's information-analytic design proposal. Bertin introduces the informal notion of the *efficiency* of a graphic based on Zipf's notion of "mental cost" as applied to visual perception [Zipf, 1935] and this is consistent with Larkin and Simon's definition of cognitive efficiency. Bertin's applies the notion of efficiency to locating a graphic design for a given data set such that "all questions [about the data set] are answered in a single instance of perception," where the set of "all

[4] Bertin's information analysis is limited to what mathematicians would refer to as *function dependency relations*. Roth et al (1989) make the point that graphics are commonly used to encode several other types of relational information. This issue is expanded in Chapter 3.

questions" is the set of "information communication" tasks considered in Bertin's work. Bertin discusses scores of other important graphic design issues such as perceptual procedure learning, decoding, salience, grouping effects, and distortion. Graphic design strategies are recommended that are consistent with experimental investigations of the same topics (reviewed below), although these works are not cited in the book.

Kosslyn (1989) provides an in-depth analysis of graphical conventions aimed at identifying conventions that lead to erroneous interpretation of graphics. Kosslyn proposes a three-fold analysis technique that considers the syntax, semantics, and pragmatics of a presentation. Kosslyn's *syntax* refers to the graphical organization of the points, lines, and areas that appear in a presentation. Graphical syntax is largely concerned with issues such as perceptual discriminability, salience, processing limitations, distortion, and continuity. *Semantics* refers to the correspondences between the "visual variables [Bertin, 1983]" used in a graphic and the domain sets of information that they are used to represent. Kosslyn's basic principle of good graphical semantics is that each visual variable used in a graphic should represent exactly one dimension of information, an issue discussed in detail in Chapter 7. The *pragmatics* of a graphic loosely describe the appropriateness of the graphic for a stated task or purpose. Graphical pragmatics considers the amount and type of information to be included in a presentation with respect to the information needs of the user, as well as choosing an appropriate format for each type of information.

An important notion advanced in early books concerned with the use of graphic presentations in business practices was that of maintaining the *integrity* of a data set when that data set is encoded graphically. Business graphics manuals argue that the preparation of a graphic should be viewed as the rigorous transformation of numerically expressed data to graphical expressions of precisely that same data. A graphic presentation should only serve to change the efficacy to which a set of possible valid inferences can be made about a data set, not affect the inferences themselves. Huff (1954) exposed many creative

graphical practices used to entice readers into drawing "creative inferences" that do not follow from more honest presentations of a data set. Huff's book, entitled *How To Lie With Statistics*, gives examples collected from books, magazines, and newspapers. Huff's examples can be categorized into two general types of "graphical lies." The first type of lie relies on the use of a graphical encoding convention for which the reader is unlikely to follow when decoding the graphic. Figure 8 shows a "crooked bar chart" in which the absolute *height* of each bar encodes a data value. The crooked bar chart succeeds when the reader is tricked into basing their comparisons between data values based on the absolute vertical *position* of the top of each bar. For example, the reader might infer that the Columbian and Brazilian tourism industries are comparable since the two bars in the graphic extend up to the same vertical position.

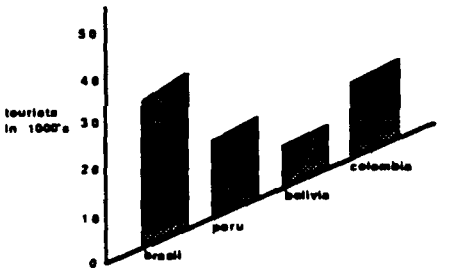
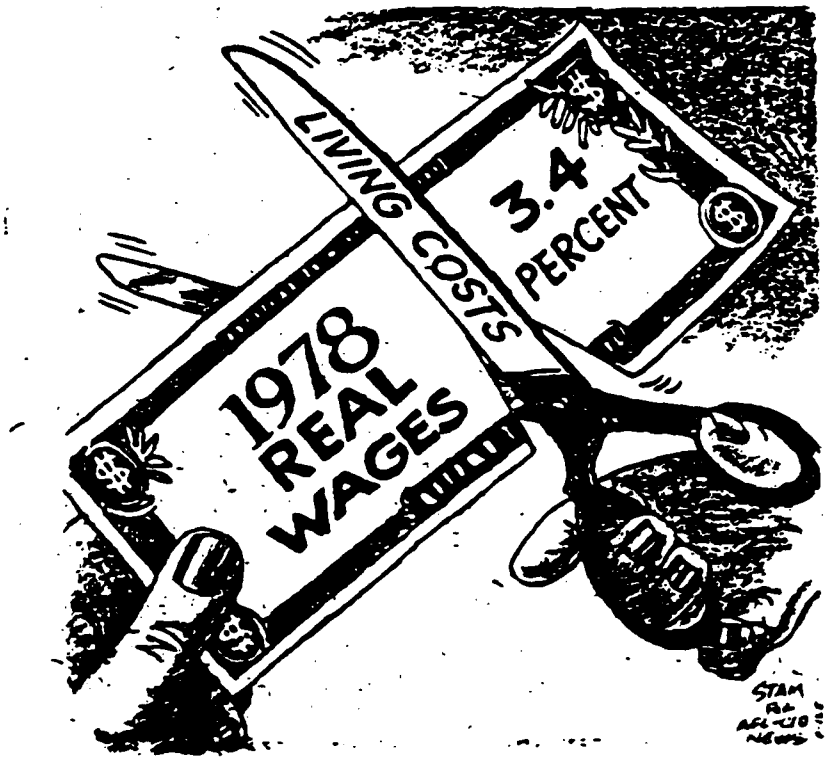


Figure 8: A (Very) Crooked Bar Chart

A second type of "lie" uses less sophisticated tactics. The graphic at the top of Figure 9 depicts a 3.4 percent rise in price levels for the year 1978. The graphic uses a scissors positioned along a dollar bill to convey the percentage of the dollar that is lost to the price increase. Unfortunately, the amount of the dollar being cut away is somewhat larger than the actual cut. The graphic at the bottom of Figure 9 shows the scissors when repositioned to reflect the actual amount of the dollar being cut away. This type of lie employs an honest encoding convention but neglects to accurately map data values between graphical and non-graphical scales. The scissors example appears in a second, more recent book on the misuse of statistics and graphics [Jaffe and Spirer, 1987].



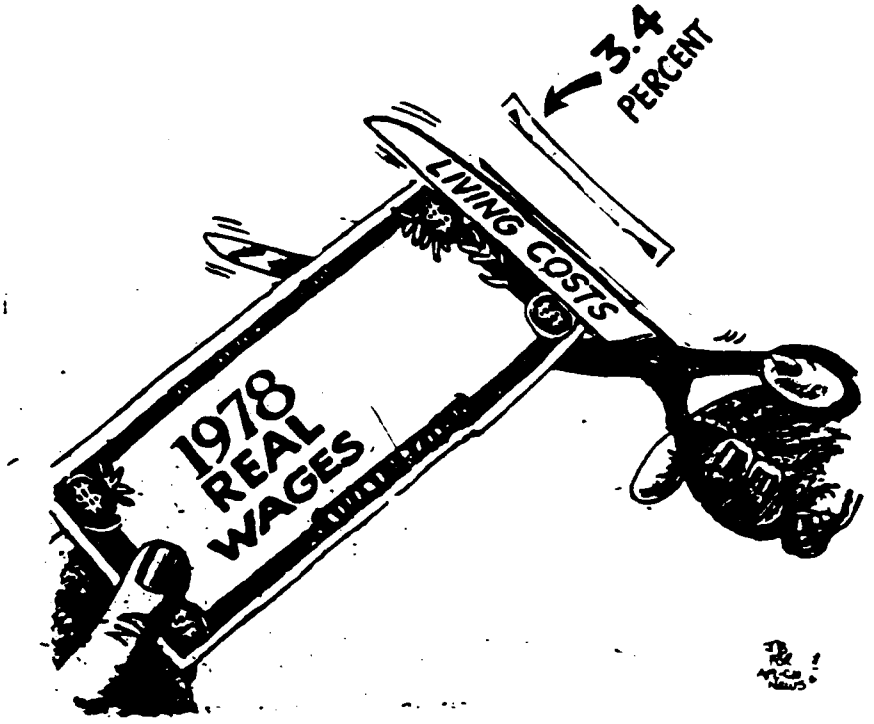


Figure 9: A Second Type of Graphical Lie.

2.3 Automated Graphic Presentation Tools

A second area of graphic design research aims to *automate* the design and rendering of graphical presentations. By formally articulating graphic design theories in working computer algorithms, this body of research aims to be more explicit about graphic design criteria, bringing the details of each design approach into plain view. Since our understanding of the issues pertaining to graphics use is growing rather than complete, automated approaches risk generalization in many areas to achieve operationality.

An early proposal for automated presentation design was Bartlett and Smith (1973) who applied facilities-allocation algorithms developed for industrial engineering problems such as plant layout [Haygood et al, 1964; Freund and Sadosky, 1967] to the problem of locating an optimal spatial layout of a set of dials and controls in an instrument panel. Bartlett and Smith's program coordinated usability parameters such as eye-travel and hand-travel distance between instruments, frequency of use for each instrument, and accuracy of acquisition.

APT [Mackinlay, 1986] is an automated graphic presentation tool that designs non-interactive graphic presentations of relational information. A significant contribution of APT was to formally characterize something that many previous investigators informally alluded to: that graphic presentations can be expressed as sentences in a formal graphical language that have the same precise syntax and semantics as propositional formalisms. The advantage of having a formalism for graphic presentations is that it provides a set of criteria for deciding the role of each visible sign or symbol placed in a graphic, and improves the integrity of a graphic presentation by using formal methods for transforming relational facts to graphical facts. APT's style of analysis for formal graphical languages has been used by nearly every graphic presentation tool designed after APT, including the one described in this dissertation. A second contribution of APT is that, unlike proposals for graphic design practices, APT designs graphics with a minimum amount of intervention on the part of the

designer. That is, APT embodies a genuinely *prescriptive* theory of how to design a graphic. However, APT's design algorithm is based entirely on an analysis of the information to be presented and does not consider the task for which a graphic is to be used. This prevents APT from directly exploiting the task-related advantages of graphics, and from creating different presentations of the same information to support different tasks.

SAGE [Roth et al, 1989] is a hybrid, text and graphics presentation system that generates explanations of changes that occur in quantitative modeling systems such as project modeling systems and financial spreadsheets. Graphic presentations are designed by SAGE in response to information queries made by the user. Through an analysis of user queries, SAGE's design of graphic presentations is sensitive to the goals of the user, taking an important step toward exploiting the task-related advantages of graphics. SAGE presently contains only a small set of primitive pre-defined task operators and is not able to fashion presentations to support complex information-processing tasks involving combinations of many primitive operators. SAGE does not make use of any theory of presentation-based problem solving when designing a presentation to support a user task.

Chang (1987, 1988, 1989) describe a family of algorithms and tools aimed at formalizing the use of pictorial icons as devices to be used for the rigorous encoding of information. Chang's research is centered around the observation that despite the representational and interpretive latitude offered by informal strategies customarily used to encode and decode iconic information, informal strategies disallow any sort of analysis of what can and cannot be formally encoded iconically, and at the same time allows rampant use of misleading or blatently dishonest graphical techniques such as those described in Huff (1954). Drawing on a set of primitive atomic icons that have pre-established meanings, Chang's systems create complex icons by applying a set of composition rules to the primitives. The meaning

of any complex icon is functionally dependent on the primitive icons used and the composition rules that were applied.

Gargan, Sullivan, and Tyler (1988) have partially implemented an adaptive response planning front-end that can present information to the user in three modalities: text, graphics, and speech. The tool is targeted for use in particular domains and relies on a formally specified domain theory describing the kinds of information to be presented in the application domain. The tool suffers two basic limitations. First, decisions made between alternative modalities are based on highly general task-independent principles (e.g., "when given graphical and textual redundancy favor graphical mode") that empirical studies of people using graphics find inappropriate. Second, decisions made about designing within each modality are centered around a set of canned presentation designs contained in the tool and purely information-theoretic criteria that describe when they are most effectively deployed (e.g., bar charts are best suited for small data sets).

CUBRICON [Neal and Shapiro, 1989] is a multi-media interface designed for a military tactical air control application. CUBRICON accepts and presents information to the user in any of four modalities: color graphics, tables, written text, and spoken language. CUBRICON relies on an extensive knowledge base representing the details of the application domain and the types of tasks that users must perform. Three basic types of decisions are made when preparing presentations for a user. First, a subset of relevant information must be selected from CUBRICON's knowledge base that is deemed germane to the current user's task. Second, a modality (or combination of modalities) must be chosen to present the relevant information. CUBRICON uses the same generalized principle that Gargan et al uses: "our system is based on the premise that graphic/pictorial presentation is always desirable." Third, within each modality CUBRICON chooses among six canned presentation formats. Presentation design criteria follow simple information-based guidelines that do not consider the details of the task being performed.

CUBRICON additionally contains an interesting technique for using pointing (deixis) to allow the user to secure references when engaged in dialogues with the system.

Integrated Interfaces [Arens, Miller, and Sondheimer, 1989] is an information presentation tool coupled with an application-specific knowledge base. This system performs two basic functions: selecting relevant information to appear in a presentation, and determining an appropriate presentation format (i.e., either tables or graphics). Relevant information is selected according to inferences drawn using the application's knowledge base (a propositional semantic net). This feature allows the tool to consider the details of the application domain when selecting information, a step not possible in generalized presentation tools such as Mackinlay's APT and the one described in this dissertation. Furthermore, since the tool is highly domain specific, many of the presentation design decisions are subsumed by existing conventions used in the application domain. For example, ships are customarily represented as icons, the position of the icon in the presentation indicating its present location in the sea. Consequently, Integrated Interfaces performs limited design once a modality has been chosen.

AIPS [Zdybel et al, 1981] accepts descriptions of information encoded using the KL-ONE [Brachman and Schmolze, 1985] knowledge representation language. AIPS matches the KL-ONE descriptions against a set of pre-defined presentation formats and chooses that format that best matches the characteristics of the data. AIPS is not able to design novel graphic presentations.

BHARAT [Gnanamgari, 1981] accepts descriptions of data sets and chooses one of bar chart, pie chart, and line chart to depict the data. The presentation format chosen is determined by the characteristics of the data: line charts are used for continuous data, pie charts for proportional data, and bar charts for all others. Like AIPS, BHARAT chooses among existing graphic designs rather than designing new ones.

VIEW [Freidell, 1982] creates graphic presentations of information about ships maintained in a naval database. VIEW's knowledge base contains information about particular users, a set of tasks for the domain, and a set of pre-defined KL-ONE descriptions of possible presentation formats. By matching users' identities, tasks, and queries against the presentation format descriptions, VIEW is able to present different graphics in different contexts. As with AIPS and BHARAT, VIEW does not design its presentation formats.

APEX [Feiner, 1985] creates graphical explanations of actions performed with physical devices in a 3-dimensional world. Explanations are created by presenting sequences of static images depicting the individual steps in an action. APEX uses hierarchical descriptions of the objects that can appear in an explanation, where each level in the hierarchy contains more detailed features of the object. A second mechanism allows APEX to determine how much detail is needed at each step and to display only that information. Since the objects that appear in explanations are highly domain-specific, they must be hand-created prior to using APEX.

COMET [Feiner and McKeown, 1990] designs coordinated text and graphics explanations of how to operate complex devices. COMET designs an explanation by first choosing a set of facts about a device and actions to be performed using the device that are relevant to an explanation. COMET chooses a presentation format (text or graphics) for each fact and action. Presentation format selection uses a set of canned decisions about which format best supports user understanding of a fact or action. These decisions were generated in advance by authors based on informal studies of user preferences. Graphic presentations are designed using a separate tool called IBIS that acts as a subsystem of COMET.

IBIS [Seligmann and Feiner, 1989] generates realistic presentations of complex devices in response to a set of communication goals for that presentation. Communication goals describe presentation characteristics of the individual components of a device that can be

manipulated to achieve different effects. For example, components can be visible or occluded in an presentation based on their relevance, highlighted to command the viewer's attention, or de-emphasized to avoid the viewer's attention.

2.4 Experimental Studies of People Using Graphics

The following surveys two areas of experimental research relevant to the topics presented in this dissertation. The first area of research is concerned with understanding the psychophysical properties of human perception and peoples' ability to perform primitive perceptual tasks. The goals of these studies are: (a) to discover ways in which perceptual encoding of information using each of the available perceptual dimensions can be advantageous to information processing and comprehension; and (b) to arrive at criteria for deciding when each type of perceptual encoding should be used. The second area of research is concerned with assessing the overall utility of graphics when used as tools to support complex information-processing tasks. The discussion below suggests that the goals of these studies are often ill-stated. Consequently, the experimental practices used are seldom consistent between studies, making interpretation and comparison of the results a difficult enterprise. One important generalization has been made in many of these studies, namely, that the utility of any presentation is largely a function of the task for which it is being used.

2.4.1 Studies of Human Visual Perception

The following studies have theoretically and experimentally explored features of human visual perception by studying human performance on primitive perceptual tasks when information is encoded using a single perceptual dimension. The goal of this analysis is to identify the task-related advantages of encoding information in each perceptual dimension.

Visual Search. Neisser (1964) was first to experimentally demonstrate the phenomenon described in Chapter 1 known as *indexing*. Neisser's experiments showed that the

identification of a target word in a list did not require that the eye be fixed on each candidate word in the list. Neisser's theory of pre-attentive processing explained previous observations that some perceptual dimensions enjoyed special properties when used in visual search tasks. For example, Green and Anderson (1956) showed that the time to locate an object of a particular color in a presentation is primarily a function of the number of objects having that color. The total number of objects in the presentation has a lesser effect [Smith, 1962, 1963; Carter and Cahill, 1979; Carter, 1982; Luder and Barber, 1984]. In a second series of experiments Neisser (1963) showed that participants, after sufficient practice, could search for any one of as many as ten items simultaneously, where the time to locate one of the ten items was no greater than the time required to locate a single target. Visual search time has been shown to depend upon data density [Carter and Cahill, 1976; Monk and Brown, 1975; Brown and Monk, 1975], target uncertainty [Monk, 1976], the dimensionality of the target item [Teichner and Mocharnuk, 1979], the proximity of the target item to the center of the presentation (the "edge effect") [Monk, 1981], visual conspicuity [Monk, 1981], workload history [Matthews, 1986], choice of presentation symbology [Remington and Williams, 1986], contrast, luminance, display duration [Teichner and Krebs, 1972], and size [Teichner and Krebs, 1972; Bloomfield, 1972]. Morawski et al (1980) and Arani et al (1984) propose general quantitative models designed to predict search times under varying search constraints and conditions.

Zeki (1978) provides physiological data supporting the notion of *functional parallelism* in visual search. Zeki argues that different areas of the visual cortex process different perceptual stimuli and that these processes can occur simultaneously at a single location. Eriksen and Hale (1955) showed that visual search could be expedited by using shape and color to redundantly encode a single dimension of information. This result suggested that visual search could simultaneously exploit spatial and functional parallelism. That is, several different perceptual operators could be applied simultaneously at multiple locations in a presentation. This hypothesis was later rejected in studies by Treisman (1977) and

Treisman and Gelade (1980) who showed that participants were unable to pre-attentively locate a target object having two unique perceptual properties.

Gould and Schaffer (1967) studied the effects of divided attention when monitoring multiple presentations. Results showed that participants could successfully monitor up to sixteen different presentations without a decrease in monitoring performance as long as the rate of change for each presentation did not exceed some threshold.

Highlighting/Luminance. Goldstein and Lamb (1967) investigated the use of single-color flashing lights to encode messages in a naval application. Results showed that participants were able to easily master a set of four messages when flash rate was used to encode the message.

Smith and Goodwin (1971) used blink coding to mark target objects in a computer display. Results showed that search time was fifty percent faster when target objects blinked at a 3 Hz rate. The data suggested no such savings when simple (non-blinking) highlighting was used. Smith and Goodwin (1972) demonstrated that this result did not hold in a text reading task indicating that the usefulness of blinking was task-dependent.

Teichner and Krebs (1972) studied the effects of luminance, duration, and area of objects in a presentation on visual detection performance. Results suggested a reciprocity between duration and luminance for small areas but no such relationship between luminance and area.

Fisher and Tan (1989) introduced qualifications on the successful use of highlighting in presentations. Results suggested that the utility of highlighting is a function of the type of highlighting, the descriptiveness of the highlighting, and the probability that users of the presentation will first attend to highlighted items. Fowler and Barker (1974) studied the

effect of highlighting in textbooks on college students' retention of text material. While no general benefits of highlighting were found, the results suggested that highlighting *improved* retention of selected text material. Highlighting was found to be most effective when readers believed that the highlighting discriminated between important and trivial information.

Color. Christ (1975) and Davidoff (1987) review experimental research pertaining to the use of color in graphic presentations. A first basic task-related advantage of using color is its utility in decreasing search time for target objects. Green and Anderson (1956) suggest that human capabilities for pre-attentively processing color allows search time for a target object to be a function of the number of objects in the presentation having precisely that color. The total number of objects in the presentation has a lesser effect [Smith, 1962, 1963; Carter and Cahill, 1979; Carter 1982, Luder and Barber, 1984]. Studies reviewed in Christ (1975) show that color is superior to other perceptual attributes such as size, shape, or brightness for supporting efficient pre-attentive search. The search advantages of color coding, however, have been shown to increase with: (1) the density of a presentation [Cahill and Carter, 1976]; (2) the number of different colors used in the presentation [Smith, 1962; Cahill and Carter, 1976]; (3) the randomness of the spatial layout of the objects in a presentation [Farmer and Taylor, 1980]; and (4) the difficulty of discrimination between colors [Farmer and Taylor, 1980; Carter, 1982]. Shontz et al (1971) and Luder and Barber (1984) report decreases in search time when color was used to redundantly encode information in a presentation. Kanarick and Petersen (1971) and Saenz and Riche (1974) found redundant color coding to be of no use in reducing overall search time but that it sometimes made possible other visual search strategies. These conflicting results suggest that the usefulness of color coding is a function of the details of the task that it is being used to support. Konz and Koe (1969) provides additional evidence of the task-related nature of the utility of color coding. Konz and Koe, in an alphabetic name card filing task, found that cards were filed most quickly when the second letter of each name

was color coded. No advantages were associated with encoding the first letter of the names.

Ware and Beatty (1988) investigated the use of the three primary colors (e.g., the red, green, and blue primaries of the television monitor) to simultaneously encode three separate discrete data dimensions in contrast to the usual strategy of mapping non-primary colors in a color space to the data values in a single data dimension. Participants viewed scatter plots representing five dimensions of information: two dimensions represented by the *x* and *y* coordinates of each point in the plane, the other three dimensions encoded using the three primary colors that together comprised the overall color of each point. In a first experiment, participants were asked to determine the presence of clusters, or collections of points agreeing along all five data (perceptual) dimensions. Response accuracy was the dependent measure. Results indicate that the use of color is reliably advantageous for this experimental task. Two further experiments measured participants' ability to make cluster judgements as each of the five perceptual dimensions were systematically varied, covering the complete space of possible color scatter plots. Results showed that in well distributed data sets the utility of the three-dimensional color coding appears as good as other encoding schemes that employ other perceptual dimensions. For data sets in which only the color dimensions vary (all points spatially clustered together), the usefulness of color decreases. The authors did not investigate or make claims about peoples' ability to identify constituent primary colors or relate them to particular data values.

No good evidence is available suggesting that color coding is useful for identification tasks [Davidoff, 1987]. Color naming tasks are especially error prone when the number of permissible colors is greater than three or four [Luder and Barber, 1984; Zwaga and Duijnhouwer, 1984]. Also, difficulty of discrimination between colors increases with the number of colors used [Bundesen and Pedersen, 1983]. Behan et al (1972) studied the problem of underwater color perception. Color also appears to impair the accuracy of size

and shape judgments when the size and shape discriminations are themselves difficult [Morgan and Alluisi, 1967]. Cuff (1973) observed that since color is actually composed of three perceptual dimensions (e.g., intensity, hue, and saturation), color-encoded data was often misread when more than one dimension was varied. Garner (1974) later showed that hue and saturation are *integral* perceptual dimensions. Integral perceptual dimensions cannot be perceived by humans independently of one another. Perceptual dimensions that can be perceived independently are called *separable*. For example, color and shape are separable dimensions [Garner and Felfoldy, 1970].

Studies of memory for color offer little support for the advantages of using color for short-term or long-term information retention. Studies show color to be less useful when short-term memory loads become high [Wedell and Alden, 1973] and of the same utility under low short-term memory demands [Alden et al, 1971]. In these cases, spatial or verbal codes are preferred. For long-term memory tasks, it appears that color information is re-encoded verbally [Paivio and te Linde, 1980]. Colors appear to be best remembered when color names are attached to them upon stimulus onset [Hendrick et al, 1968]. Loftus (1977), in a study of eyewitness testimony, showed that memory for the color of an object can be contaminated when participants are exposed to misleading verbal information about the color of that object.

Albers (1963) demonstrates the effect known as *color interaction* that occurs when objects of one color are embedded within objects of another color. Color interaction can produce two interesting context-sensitive effects: (1) identical colors may seem different; and (2) different colors may seem identical. Tufte (1990) gives an example of each of the two color interaction effects (pp. 92-93).

Shading. Shading offers many of the same search advantages as does color. One important difference between shading and color is that, since shading consists of a single

dimension (*luminance* or *grey scale*) there are fewer distinguishable shades than there are distinguishable colors. Hence, the number of categories that can be encoded with shading is somewhat smaller. Other issues complicate the use of shading. It is a well known phenomenon of psychophysics that perception of equal grades of darkness do not coincide with equal grades of ink/area. Williams (1956) proposed a grey scale whose increments were claimed to agree with peoples' grey scale judgements. Participants in a study by Jenks and Knos (1961) compared five different grey scales, one of them anonymously the Williams scale, and decided that the Williams scale was the most reliable. Jenks and Knos further add that varying certain characteristics of the texture used in a shaded region, such as size and arrangement of dots, can cause shades to appear out of order. Consequently, care must be taken when texture is used together with shading.

Texture. Julesz (1981) investigated participants' ability to pre-attentively process texture when used as a data-encoding perceptual dimension. Julesz found that certain types of textures composed of elements called *textons* could mediate pre-attentive processing. Julesz's textons include color, elongated shapes of specific sizes, and orientations. Since the textons described by Julesz are themselves perceptual dimensions that have been shown to have pre-attentive processing properties (i.e., indexable), these results are consistent with Neisser's earlier observation that some perceptual dimensions enjoy indexing properties when perceptually treated as other dimensions understood to be indexable.

Size. Bloomfield (1972) demonstrated that the time to locate an object in a presentation is in general inversely proportional to the size of that object. Teghtsoonian (1965) studied the accuracy to which participants performed area judgements. Teghtsoonian found area judgements to be reliably erroneous. The result of Teghtsoonian's study was to propose a correction formula that could be applied to any graphic that used area to encode quantities. The correction formula states that the perceived area is typically the actual area raised to a power of 0.8. Stevens (1974) describes a similar formula that can be applied to lengths

and volumes as well as areas. The length and volume correction formulas are identical to Teghtsoonian's formula excepting the exponent which is {0.9 ~ 1.1} for length judgements and {0.5 ~ 0.8} for volume judgements. Stevens reviews a previous formula known as Weber's Law which specifies a quantity that must be added to a physical magnitude in order to insure that accuracy of judgement occurs with a given probability, *p*. Flannery (1956) studied errors made in area and diameter judgements for circles. Flannery found regularity in the errors in both types of judgements and proposed a correction formula for circles similar to Stevens' Law.

Luria and Kinney (1970) and Vernoy (1989) studied the problem of underwater perception of size and distance when a diving mask is worn. Divers typically overestimate the size of unknown objects and underestimate the distance to these objects. Correction formulae are proposed by the studies.

Shape. Studies of visual search suggest that some characteristics of shape are perceived pre-attentively. For example, Neisser (1964) showed that participants could identify letters and words pre-attentively based on the shape of the word. Although the relevant shape characteristics cannot be positively identified from the existing data, three characteristics have been proposed in Ullman (1984): (a) curvature; (b) orientation; and (c) number of terminating points. Larsen and Bundesen (1978) demonstrated that search time for matching shapes within a figure increased linearly when the size or scale of one object in the matched pair was increased (preserving shape).

Icons. Macdonald-Ross (1977) and Barnard and Marcel (1987) review research pertaining to the use of icons. A notion central to all theories of icons recognition is that the recognition process invokes real-world knowledge [Marr and Nishihara, 1978]. Barnard and Marcel point out that abstract icons may invoke conceptual-level referents while more realistic icons may invoke knowledge about specific instances. They also point out two

important limitations on the use of icons. First, the interpretation of icons may differ widely among occupations or cultures [Cahill, 1976; Neurath, 1936; Hudson, 1968]. Second, many abstract concepts have no natural depictive form (e.g., sound, logical relationships). Easterby (1970) found that increasing the amount of detail in a set of icons used to represent machine parts did little to enhance machinists' ability to recognize the symbols.

Pictorial charts use icons to represent a set of entities and quantitative facts pertaining to the entities. Brinton (1916) introduced the simple technique of encoding quantitative attributes of an entity by varying the size of the icon used to represent that entity. Neurath (1936) flagged this convention as erroneously interpretable and suggested that multiple instances of an icon be used to encode quantities. Neurath (1944) pointed out several other issues pertaining to interpretation accuracy such as horizontal and vertical alignment of icons, and how different alignments could entice readers into drawing different inferences. Macdonald-Ross (1977) extensively reviews the development of the pictorial chart.

Connectivity. Krohn (1983) studied subjects' performance time and accuracy when flowcharts were used to support a decision task. The directional orientation, the number of alternatives leading from each decision box, and the decision task complexity were manipulated as independent variables. The results showed that performance was best when flowcharts were oriented consistently with reading patterns (left to right). The number of alternatives issuing from each decision box had no effect on performance.

Tables. Ehrenberg (1975) and Chakrapani and Ehrenberg (1976) informally studied the use of tabular presentations to support information-processing tasks. These studies propose the following set of simple design principles that can help improve the effectiveness of a table:

1. Round numbers to two significant digits to facilitate mental arithmetic.
2. Provide row and column averages.
3. Column comparisons are performed more easily so use columns for the most important comparisons.
4. Order rows and columns by size of of numbers, not by alphabetical order of labels.
5. Columns and rows should be compactly set, not spaced out in order to fill the page.

Note that the design principles are sensitive to the types of cognitive design issues addressed in this dissertation, namely, reducing mental computation and streamlining visual search. Efficacies of principles 2 and 4, however, may only coincidentally occur when the table is used for a task that makes use of row and column averages, or that requires the user to search for particular attribute (column) values. Principles 2 and 4 may work against the user when performing tasks that use other types of quantities or that require the user to search on row elements.

Wright (1968, 1972) conducted a number of experiments concerned with differences in accuracy of interpretation between tables and graphs. A long-standing criticism of graphics was that readers would often make mistakes when looking up data values. Proponents of this argument suggested that tables be used whenever accuracy was important [Brinton, 1939; Tufte, 1983]. Wright's work with tables demonstrated an important phenomenon that caused the "tables-for-accuracy" argument to be qualified. Wright summarizes her results as follows:

> "If the user of a table is required to carry out operations other than *search* and *read*, the number of competent users is markedly reduced. Tabulation schemes requiring synthetic or analytic operations by the user seem to buy economy of space at the cost of comprehension [Wright, 1973]."

2.4.2 Studies of Complex Task Performance Using Graphics

Investigators in Human Factors and Management Information Systems (MIS) have studied the use of graphics as an aid to the performance of complex information-processing tasks. Complex tasks involve the performance of many primitive perceptual tasks such as those discussed in the previous section. The goal of these studies is to establish the utility of

graphic presentations as tools to support information-processing and decision-making tasks. The studies have largely failed to accomplish this goal, reporting mixed and often contradictory results. One reason for these mixed results may be the lack of consistency among the experimental methods used. Inconsistency among methods makes generalizations of individual results difficult, and comparisons between studies less meaningful. Methodological inconsistencies among studies occur in at least six forms. First, there is no general agreement as to what *dependent measures* are important when testing users' performance with graphics. Two popular choices are: (1) task performance time; and (2) accuracy. Studies typically choose one or the other but seldom measure both. Second, *experimental procedures* frequently differ in the instructions given to participants. A basic problem arises with this inconsistency in terms of the speed-accuracy trade-off in subjects' task performance. Subjects who are told that they are being timed often trade away accuracy for the purpose of completing the task more quickly. This makes accuracy measurements difficult to compare. Subjects who are told to strive for accuracy often use more time than necessary, sometimes reviewing their work several times before reporting an answer. Subjects who are told to maximize both time and accuracy may exhibit performance characteristics different from any group. Third, some studies include a *task analysis* describing in detail the activities that the participants are likely to be performing while being measured [Lusk and Kersnick, 1979]. Drawing strong conclusions from data is often risky when an understanding of the procedure that participants followed is not available [Newell, 1973]. Fourth, very few studies include an *analysis of the presentation* that participants use to perform the experimental task [Lusk and Kersnick, 1979]. A basic assumption underlying these studies is that the features of a presentation can effect users' performance yet few studies pay any attention to the features of the particular presentations they test. Some studies do not even provide the reader with examples of the experimental graphics used [Dickson et al, 1986]. Fifth, some studies attempt to control the level of *user skill or practice* with the experimental task or stimulus materials used. A common technique for measuring individual differences is the Embedded Figures Test which has

been argued to correlate well with spatial ability [Witkin et al, 1971]. More recent studies use pre-tests to determine participants' prior knowledge of the subject matter [Hegarty and Just, 1988].[5] When no attempt is made to measure individual differences in skill or knowledge it should seldom be surprising that a person having experience with a particular presentation should outperform another person using a presentation that is unfamiliar to them. Sixth, and most important, many studies are inconsistent in the *conclusions* they reach. Many studies find tabular presentations of a particular type of information to be superior, while others find no significant differences, while still other find graphics to be superior. The implications of this finding are discussed below.

Cleveland and McGill (1984) tested the accuracy to which subjects were able to perform a simple information extraction task when data values were encoded using each of the perceptual dimensions. Information was encoded in separate graphic presentations, each presentation using a single perceptual dimension such as shape, color, size, spatial position, etc. Participants' accuracy scores were used to rank the perceptual dimensions in order of their observed interpretation accuracy. The rankings were used as a means of explaining why some graphics appear to be more successful than others, namely because they use more accurately interpreted perceptual dimensions.

Lusk and Kersnick (1979) investigated the effect of presentation format on task performance accuracy from a task-analytic perspective. Participants answered twenty questions using two different presentation formats: tables and graphs. For each combination of presentation format and question, the set of problem-solving operators required to answer the question was enumerated. Response accuracy was the dependent measure. Lusk and Kersnick found that in no case did the use of a graphic significantly improve task performance over a tabular presentation of the same information.

[5] Other pre-tests are sometimes used to insure normal or corrected vision [Clancey and Hoyer, 1987], and accuracy of color perception [Ware and Beatty, 1988].

Lucas (1981) asked participants to perform a decision task using alternative tabular and graphic presentations. Response accuracy was the dependent measure. No detailed analysis was performed on either the task or the presentations. Lucas found that participants performed better with graphic presentations than with tabular presentations appearing on the computer screen. Surprisingly, participants performed best of all using hard copy tabular presentations (unfortunately, hard copy graphics were not tested)!

Tullis (1981) asked telephone company employees to use four different presentations of information about diagnostic test results performed on a telephone line. The four presentations were: (1) a narrative description of the test results; (2) a structured text (tabular) presentation; (3) a black-and-white graphic presentation; and (4) a color graphic presentation. The experimental task was duplicated in two separate sessions to investigate the effects of practice. Task performance times did not differ significantly between any two conditions for the first session. Significant differences were observed between structured and narrative texts during the second session. In no case did a graphic result in superior performance over the structured text.

Dickson, DeSanctis, and McBride (1986) conducted three experiments to test the effect of presentation format on readability, interpretation accuracy, and decision quality. The first experiment compared tabular and bar chart presentations for a set of simple information extraction and comparison task. Results indicated no significant differences between the two presentations. The second experiment showed line graphs superior to tables for a more sophisticated decision task. The third experiment showed that graphics were better for communicating a "message" with data when the message was simple. The authors draw three conclusions: (1) "generalized claims of superiority of graphic presentations are unsupported;" (2) "the effectiveness of the data presentation format is largely a function of the characteristics of the task at hand;" and (3) "impressions gleaned from one-shot studies

of the effectiveness of the use of graphs may be nothing more than situationally dependent artifacts.

Benbasat, Dexter, and Todd (1986) studied participants' performance on the Brand Manager's Allocation Problem (a well understood experimental MIS task) using tabular, graphic, and combined tabular and graphic presentations. The dependent measures were response time and decision quality. Results showed that the use of color graphics led to a ten percent improvement in decision quality but no significant difference in response time. No explicit instructions were given to participants indicating the relative importance of decision time vs. decision quality.

Benbasat and Dexter (1986) studied the effect of presentation format on task performance time and decision quality under varying time constraints. Participants used tabular, graphic, and combined tabular and graphic presentations to perform the Brand Manager's Allocation Problem under five and fifteen minute time constraints. Participants were given no explanation of the relative importance of performance time and accuracy. For the fifteen minute treatment, the color graphic presentation group completed the task more quickly than any other group but showed significantly lower decision quality than the combined table/graphic group. For the five minute treatment, no significant differences were found among presentation format for performance time but the combined table/graphic group produced significantly better decisions. Participants' answered ten questions in evaluation of the different presentations and decided that the combined table/graphic presentation was superior.

Benbasat et al (1986) conducted a less rigorous study of the use of color in presentations of information used for managerial decision-making tasks. The authors conclude that color is primarily useful in the early stages of a decision task and when time constraints are introduced. However, these results have little explanatory power and it is possible that

more detailed analyses such as those reviewed in Davidoff (1987) can account for the observations more concretely.

Other interesting studies address the use of graphic presentations in analogical reasoning [Beveridge and Parkins, 1987], the utility of flowcharts for understanding procedural instructions [Kammann, 1975; Krohn, 1983], the use of maps for terrain visualization [Barsam and Simutis, 1984], the use of face caricatures to encode data dimensions [Jacob et al, 1976], the use of holograms as training and job aids [Frey and Eichert, 1978], and the use of Gantt charts in job-shop scheduling [Gibson and Laios, 1978].

Jarvenpaa and Dickson (1988) review the findings of thirty-three independent studies that assess the utility of graphics, including the MIS studies discussed above. Jarvenpaa and Dickson show that sorting the studies in a tables vs. graphics fashion yields two groups of commensurate size, one concluding that tables are superior, the other arguing for graphics. It is suggested that some other criteria is needed to explain the mixed results. A criteria is introduced that additionally considers the nature of the task for which a presentation is used to support. A preliminary, coarse-grained classification of task types is proposed. This informal model accounts for some but not a significant amount of the variance between studies. The authors make two recommendations. First, future studies should adopt a more fine-grained task focus and a more serious taxonomic study of task types or task characteristics must be undertaken. Second, "the results of any graphics study can be interpreted solely as a function of the task," and that "extrapolation of results in one task activity with those of another is inappropriate unless the researcher also considers the characteristics of each task."

Figure 10 graphically summarizes important characteristics of the reviewed MIS studies of graphic presentation utility for complex information-processing tasks.[2]

[2]A shaded entry in the columns for TASK ANALYSIS, DISPLAY ANALYSIS, and INDIVIDUAL DIFFERENCES in Figure 10 indicate a "no" value, unshaded entries indicate "yes."

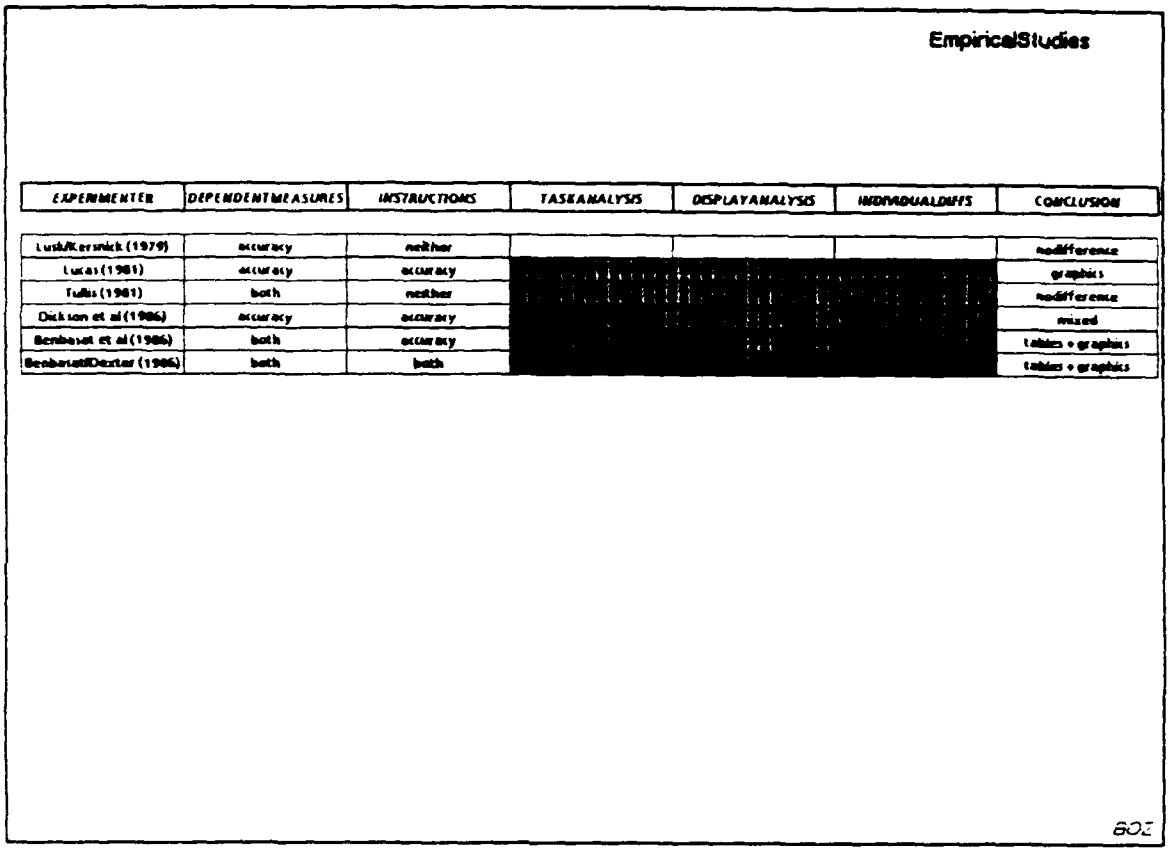


Figure 10: Experimental Studies of Complex Task Performance.

2.5 Information-Processing Models of Graphic Design and Use

Larkin and Simon's work [23] was first to study the utility of graphics from a cognitive science perspective. Larkin and Simon built detailed cognitive simulations of human task performance, each simulation performing a task using informationally equivalent logical and graphic representations of a set of data. Larkin and Simon's analysis yielded two general ways in which graphic presentation-based procedures could be performed more efficiently by humans: (1) by allowing users to substitute quick perceptual operators for more demanding logical operators; and (2) by reducing search for needed information.

Several more recent studies have investigated other cognitive utilities of graphic presentations. Hegarty and Just (1988) studied the use of realistic diagrams in understanding complex machines. Fallside (1988) investigated how learners make use of

animated diagrams when understanding complex machines. Koedinger and Anderson's work (in press) suggests that students are better able to learn and apply the rules of geometry when the rules are explained to them in terms of the way they can be applied to particular configurations of lines and angles in geometry diagrams.

Ohlsson (1987) devised three interactive illustrations used for teaching proportional reasoning concepts to children. Ohlsson formally analyzed the mathematical objects and operations pertaining to proportional reasoning tasks and devised a graphic presentation and set of perceptual inferences and graphical manipulations that formally correspond to the proportional reasoning concepts. Maintaining one-to-one correspondences between the graphical activities performed using the interactive presentations and the math concepts to be learned allows children to explore the semantics of abstract mathematical concepts using concrete examples before learning the details of an abstract formalism and notation. Ohlsson's idea of designing presentations that feature correspondences between formal mathematical reasoning steps and perceptual and graphical operations performed using a graphic presentation is central to the research described in this dissertation.

Goldenberg (1988) conducted informal studies of differences in interpretation of function graphs between mathematically naive students and mathematically knowledgeable adults. Goldenberg argues that the interpretation of a graph is a function of the specific mathematical knowledge and procedures that the user of the graph possesses. Three implications follow from this argument: (1) it is a false assumption that graphs of functions are inherently more accessible to students than other types of representations; (2) thoughtless use of graphs in instruction may serve to obscure mathematical concepts that are already difficult to teach; and (3) graphics that benefit from a close consideration of students' knowledge and instructional goals may help students access mathematical concepts obscured by cumbersome notations. Goldenberg (1989) outlines a graphic-based cirriculum for teaching mathematics concepts through the use of fractal diagrams.

2.6 Conclusions

The related research findings serve to corroborate three basic points made in the Introduction.

(1) The utility of any presentation of information is a function of the task for which that information is to be used.

> "In fact, the experiments suggest that the effectiveness of the data presentation format is largely a function of the characteristics of the task at hand [Dickson et al, 1986]."

(2) The design of any presentation of information should begin with a consideration of the task to be supported. Decisions about how to encode information in the presentation should be focused on choosing those conventions that best support the specific activities that the user must perform. Design criteria should be as articulate and formal as possible.

> "One way to systematically develop a program of empirical research is to consider how one would program a computer to emulate an expert human graph designer [Kosslyn, 1985]"

> "The benefits of graphics are limited to reducing decision making time but only when the graphical report has been designed to directly assist in solving the task [Benbasat et al, 1986]."

> "We submit that the guidelines must take into account the nature of the task supported. Several graphics experiments over the years have provided support for this contention that the effectiveness of an information presentation is highly dependent on, or sensitive to, the task being performed [Jarvenpaa and Dickson, 1988]."

> "The nature of the task to be performed with a graphic presentation is critical for decisions concerning the use of colour [Davidoff, 1987]."

(3) Empirical studies that attempt to establish the utility of a presentation format must explicitly take into account the experimental task performed and the particular presentations used.

> "... we should look for more attention to the purpose of the graphic device, better quality stimulus materials, and better definition of the target audience [Macdonald-Ross, 1977]."

"(1) the results of any graphics study can be interpreted solely as a function of the task, and, (2) any comparison or, (3) extrapolation of results in one task activity with those in another is inappropriate unless the research also considers the characteristics of each task [Jarvenpaa and Dickson, 1988]."

CHAPTER 3

TASK-ANALYTIC DESIGN OF GRAPHIC PRESENTATIONS: AN AUTOMATED APPROACH

This chapter describes BOZ, an automated graphic design and presentation system that creates graphic presentations of information customized to the requirements of specific tasks. BOZ works by analyzing the information-processing tasks that information presentations are needed to support. BOZ's task-analytic approach makes direct use of the theoretical advantages of graphic presentations developed in Chapter 1. Figure 11 illustrates BOZ's basic functionality.

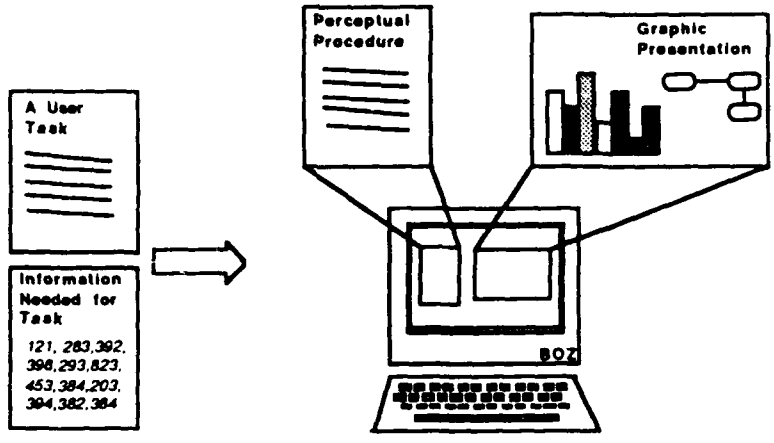


Figure 11: Overview of BOZ

BOZ requires two things as input: (1) a description of a task to be performed by a human user without the benefit of a graphic presentation; and (2) the information that the user will need to complete the task. Descriptions of tasks and information must presently be prepared by hand in advance and submitted to BOZ. BOZ produces two things as output: (1) a graphic presentation that best supports the task; and (2) a set of instructions indicating

how to use the graphic presentation to successfully complete the task. This set of instructions takes the form of a step-by-step description of the perceptual inferences the user must perform when doing the task. In this manner BOZ attempts to design an efficient perceptual way of allowing the user to obtain the same results achievable via performance of the original task. Stated another way, given a representation-independent description of a task, BOZ searches a space of graphical representation schemes looking for ways to instantiate the generalized task within the context of a particular graphic presentation. BOZ attempts to single out that particular graphic design that affords the user the opportunity to accomplish the task in the most efficient way.

A **logical task description language** allows the user of BOZ (e.g., interface designer, cognitive scientist, etc.) to describe the information-processing task that s/he wishes to design a graphic presentation to support. This language is used to enumerate the individual problem-solving steps (logical operators) that are required for a user to complete a task without the benefit of any information display.

A **perceptual operator substitution** component considers each operator in a logical task description looking for way to substitute perceptual operators in place of logical operators when the operators can be shown to produce the same output given the same input. BOZ contains a catalog of perceptual operators describing problem-solving steps performed within the context of a graphic. Perceptual operator substitution is the mechanism used to reduce the amount of mental computation performed by the human user when performing a task. Several perceptual operators typically qualify as substitutes for each logical operator, yielding a set of possible perceptual procedures.

A **perceptual data structuring** component examines the information manipulated by each logical operator and determines how information shared by several operators should be collected together to form complex graphical objects, and how unrelated information can

be partitioned into distinct presentations. Perceptual data structuring is one mechanism used to minimize the amount of time the user spends searching for information in a graphic. The perceptual data structuring component determines the optimal grouping and distribution of information within a graphic. The perceptual data structuring component does not determine how the information is to be graphically encoded in the presentation.

A **perceptual operator selection** component chooses a single perceptual operator to substitute each logical operator in a task description thus arriving at a single perceptual procedure to be performed by the user. The first criteria for perceptual operator selection is how efficiently and accurately each perceptual operator is likely to be performed by human users. Selecting each particular perceptual operator also decides the way that the information manipulated by that operator (and related operators) must be graphically encoded in a presentation. A second criteria for operator selection is choosing a complete set of perceptual operators that results in a set of graphical encodings that can be combined according to the specification produced by the perceptual data structuring component. The results of applying the perceptual operator selection component are detailed descriptions of a single perceptual procedure and an accompanying graphic design that supports the performance of the perceptual procedure.

A **graphic presentation rendering** component translates logical facts into graphical facts and displays them on the computer screen in the format specified by the graphic design produced by BOZ.

The remainder of this chapter describes BOZ's automated graphic design approach in detail. To help illustrate how BOZ works, a running example will be developed throughout the discussion. The running example further shows how BOZ can be used to analyze a real-world task and design a perceptual procedure and accompanying graphic presentation that help users accomplish the same task more easily. In the example, a graphic

presentation will be designed to support the following task pertaining to making a reservation on an airline flight:

> *Find a pair of connecting flights that travel from Pittsburgh to Mexico City. You are free to choose any intermediate city as long as the layover in that city is no more than four hours. Both flights that you choose must be available. The combined cost of the flights cannot exceed $500. Find an empty seat on each flight.*

3.1 Logical Task Description Language

The first component of the task-based design methodology is a means of making explicit the information-processing activities that a graphic presentation is needed to support. Several task description languages exist in the literature, two of them targeted specifically for use in designing user interfaces [Payne and Green, 1986; Card, Moran and Newell, 1983]. Given the particular use of a task description language intended here, two issues guided the decision to build a new task language over choosing one of the existing techniques. First, the task language should be *executable*. Task languages such as TAG [Payne and Green, 1986] and GOMS [Card, Moran, and Newell, 1983] require that the designer manipulate the formalism by hand. If the designer wishes to build a working simulation of a task, the task description must be re-encoded using a second formalism such as a production rule language.[1] Second, the specific features of other task languages do not map well onto BOZ's task specification requirements. For example, GOMS contains a mechanism for describing how users choose which operators to apply when performing a procedure, or which procedure to apply to a given task (i.e., selection rules), an issue not addressed by BOZ. An important feature missing in GOMS and ETAG is a notation powerful enough to describe the domain sets of information manipulated by a task.

[1]Chapter 4 shows how logical procedures described using the task language and perceptual procedures derived using the perceptual procedure derivation algorithm can be directly used as simulations (requiring no human intervention) for generating quantitative predictions about the utility of a presentation with respect to a task.

BOZ's task description language contains two basic components: (a) a notation for describing logical procedures; and (b) a notation for expressing relational information manipulated by a logical procedure.

Logical procedure definitions are similar to programs in conventional programming languages such as Pascal. Every logical procedure contains two parts: (a) a set of logical operator definitions; and (b) a main body. A *logical operator (LOP)* is composed of an operator name, a list of arguments taken as input to the operator, and a single relation that the operator computes. Logical operators occur in two forms. A *search operator* uses one of the three meta-commands: ASK, TELL, and RETRACT to query, assert, and remove logical facts from a simple database of logical facts. *Relational facts* contain a single predicate followed by any number of arguments. *Predicates* describe relations between two or more logical arguments. *Arguments* are variables that can be assigned relational values drawn from the collection of domain sets (explained below) defined for a task. In LOP definitions, arguments are indicated by the name of a domain set enclosed in brackets (i.e., "< >"). Arguments may be instantiated or uninstantiated. *Uninstantiated arguments*, i.e., variables that have not yet been assigned a value, are capitalized. *Instantiated arguments* are variables that were previously uninstantiated but have since been assigned a relational value. Instantiated arguments appear in lower case. Note that the same argument may be uninstantiated in one clause, be assigned a value, and appear in lower-case form (i.e., instantiated) in a later clause. A *computation operator* describes a computation performed on a set of logical arguments using one of a set of pre-defined arithmetic or logical functions such as PLUS, DIFFERENCE, TIMES, QUOTIENT, AND, OR, NOT, etc. Note that only instantiated arguments may appear in a computation operator.

The following example describes two logical operators in the airline reservation procedure. The two operators determine the departure time of an airline flight (search operator), and the layover between two flights (computation operator), respectively:

```
(NLAMBDA determineDeparture (<flight> <DEPARTURE>)
  (ASK (Departure <flight> <DEPARTURE>)))

(NLAMBDA computeLayover (<departure> <arrival> <LAYOVER>)
  (DIFFERENCE <departure> <arrival> <LAYOVER>))
```

The key word NLAMBDA is used to denote a logical operator. The lists (flight DEPARTURE) and (departure arrival LAYOVER) are the sets of arguments that the two operators receive as input. The ASK predicate states that a list of facts should be checked to see if the predicate that follows can be shown to be true: namely if there exist a fact expressing the departure time of the flight. The clause (DIFFERENCE departure arrival) specifies that the pre-defined subtraction function is to be computed given the values departure and arrival, and the variable LAYOVER is to be instantiated with the result.

The *main body* of a logical procedure is an ordered sequence of calls to the set of defined logical operators. To express control information, the main body of a logical procedure may additionally contain any of the following control constructs: **while-do**, **for**, **repeat-until**, and **if-then**.

To illustrate the use of the task language, Figure 12 shows a complete procedure description for the airline reservation task.

```
(TASK airlineReservation

(DOMAINSETS
 (flight NOMINAL 50)
 (origin NOMINAL (pit hou dal ord alb mex gdl qto paz bga))
 (destination NOMINAL (pit hou dal ord alb mex gdl qto paz bga))
 (departure QUANTITATIVE 100)
 (arrival QUANTITATIVE 100)
 (layover (departure arrival))
 (cost QUANTITATIVE 500)
 (availability NOMINAL (ok full))
 (seat NOMINAL 7200)
 (seatnumber ORDINAL (1A 1B 1C 1D 1E 1F ... 24A 24B 24C 24D 24E 24F))

(LOPS
 (NLAMBDA findFlightWithOrigin (<FLIGHT> <origin>)
  (ASK (Origin <FLIGHT> <origin>)))
 (NLAMBDA findDestination (<flight> <DESTINATION>)
  (ASK (Destination <flight> <DESTINTATION>)))
 (NLAMBDA landsInDestinationCity? (<destination> <destination>)
  (EQUAL <destination> <destination>)))
 (NLAMBDA available? (<flight>)
  (ASK (Availability <flight>)))
 (NLAMBDA determineDeparture (<flight> <DEPARTURE>)
  (ASK (Departure <flight> <DEPARTURE>)))
 (NLAMBDA determineArrival (<flight> <ARRIVAL>)
  (ASK (Arrival <flight> <ARRIVAL>)))
 (NLAMBDA computeLayover (<departure> <arrival> <LAYOVER>)
  (DIFFERENCE <departure> <arrival> <LAYOVER>))
 (NLAMBDA connecting? (<departure> <arrival>)
  (GREATERP <departure> <arrival>))
 (NLAMBDA layoverLessThanX? (<layover> <layover>)
  (LESSP <layover> <layover>))
 (NLAMBDA determineCost (<flight> <COST>)
  (ASK (Cost <flight> <COST>)))
 (NLAMBDA addCosts (<cost> <cost> <COST>)
  (PLUS <cost> <cost> <COST>))
 (NLAMBDA costLessThanX? (<cost> <cost>)
  (LESSP <cost> <cost>))
 (NLAMBDA findSeat (<flight> <SEAT>)
  (ASK (Seat <flight> <SEAT>)))
 (NLAMBDA emptySeat? (<seat> <availability>)
  (ASK (Availability <seat> <availability>)))
 (NLAMBDA findSeatNumber (<seat> <SEATNUMBER>)
  (ASK (Seatnumber <seat> <SEATNUMBER>)))

(PROCEDURE
   (while (findFlightWithOrigin FLIGHT 'pit) do
    (if (available? flight) then
     (findDestination flight LAYOVERCITY)
     (determineArrival flight ARRIVAL)
     (while (findFlightWithOrigin CONNECTING layovercity) do
        (if (available? connecting) then
         (findDestination flight FINALDESTINATION)
         (if (landsInDestinationCity? finaldestination 'mex) then
          (determineDeparture connecting DEPARTURE)
          (computeLayover departure arrival LAYOVER)
          (if (and (connecting? departure arrival)
                   (layoverLessThan4? layover)) then
           (determineCost flight COST1)
           (determineCost connecting COST2)
```

```
(addCosts cost1 cost2 TOTAL)
(if (lessThan500? total) then
  (repeat
    (findSeat flight SEAT1)
  (until (emptySeat? seat1)))
  (findSeatNumber seat1 SEATNUM1)
  (repeat
    (findSeat connecting SEAT2)
  (until (emptySeat? seat2)
  (findSeatNumber seat2 SEATNUM2)]
```

Figure 12: Logical Airline Reservation Procedure

Logical facts are used to describe relational information manipulated by a logical procedure. Logical facts state relations between values drawn from one or more domain sets. *Domain sets* are the information types that define the universe of discourse for a task. Domain sets associate a name, a type of information, and a (possibly infinite) set of data values of that type that can appear in a logical fact. Three types of domain sets are allowed in the present version of BOZ: *quantitative*, *nominal* and *ordinal* [Stevens, 1946].

The airline reservation procedure manipulates information from the domain sets specified in the top portion of Figure 12. Figure 13 shows a set of logical facts that describe two airline flights.[2]

```
(flight flight117)                    (seat flight117 seat001)
(flight flight239)                    (seat flight117 seat002)
(origin flight117 pit)                .
(origin flight239 pit)                .
(destination flight117 hou)           .
(destination flight239 alb)           (seatnumber seat001 1A)
(availability flight117 ok)           (seatnumber seat002 1B)
(availability flight239 ok)           .
(cost flight117 179)                  .
(cost flight239 219)                  .
(departure flight117 10)              (availability seat001 ok)
(departure flight239 8)               (availability seat002 full)
(arrival flight117 12.83)             .
(arrival flight239 12)                .
```

Figure 13: Factbase for the Airline Reservation Task

[2] The complete list of seats for the flights is quite voluminous and is abbreviated here.

3.2 Perceptual Operator Substitution

Perceptual operator substitution is the technique used to insure that a graphical display best exploits the first task-specific advantage of graphics: that users can substitute efficiently performed perceptual inferences in place of more demanding logical inferences. The perceptual operator substitution component considers each logical operator appearing in a logical task description looking for ways of substituting perceptual operators in place of logical operators when the logical and perceptual operators can be shown to be equivalent. The perceptual operator substitution component produces a set of perceptual operators that can potentially serve as substitutes for each logical operator. Decisions about which particular perceptual operator to substitute for each logical operator are subject to further design criteria described in Section 3.4.

Perceptual operator substitution relies on two important components: a *catalog of perceptual operators* that describes information-processing activities that occur within the context of a graphical display; and a *substitution algorithm* that considers each logical operator in a task and searches the catalog of perceptual operators for those perceptual operators that compute the same function as the logical operator. Since there are often several perceptual operators that qualify as substitutes for a logical operator, the perceptual operator substitution component produces a set of possible perceptual procedures equivalent to the logical procedure.

3.2.1 A Catalog of Perceptual Operators

Perceptual operators (POPs), analogous to logical operators (LOPs), characterize human information-processing activities performed within the context of a graphic presentation and whose performance depends on the use of a graphic presentation. Perceptual operators describe visual computation or search performed using graphically expressed information. For example, judging the distance between two objects in a presentation, and locating an object having a particular color are examples of perceptual operators.

Perceptual operators are organized around a set of *primitive graphical languages* available to the designer of a graphic presentation. Primitive graphical languages formally capture the graphic designer's resources for representing information graphically. The notion of a formal graphical language is due to the work of Bertin (1983) and Mackinlay (1986). The set of primitive graphical languages used by BOZ is shown in Table 2. The parenthesized numbers in Table 2 indicate an upper limit on the number of distinct values that can be practically encoded in a single graphic presentation using each primitive graphical language. The primitive graphical languages used by BOZ are separable [Garner, 1974]. A *separable* graphical language can be processed by humans independently, that is, the interpretation of a separable language is not affected by the presence of other graphical languages. For example, Color and Horizontal Position are separable because peoples' ability to determine the color of a graphical object is not affected by the horizontal position of that object. The color of an object can be determined without attending to horizontal position, and the color perceived is not affected by horizontal position.[3]

Table 2: The Primitive Graphical Languages

Horizontal Position (100)	Color (12)
Vertical Position (100)	Labels (∞)
Height (50)	Line Thickness (3)
Width (50)	Line Dashing (2)
Line Length (50)	Shape (5)
Area (10)	Visibility (2)
Shading (4)	Tabular (∞)
Connectivity (8)	

Associated with each primitive graphical language is a set of perceptual operators (POPs) that are admitted when that primitive graphical language is used to encode information in a graphic. For example, if we elect to use the Horizontal Position language to encode information in a graphic we admit a family of perceptual operators (POPs) such as

[3]Graphical languages that do affect one another are called *integral* graphical languages [Garner, 1974]. For example, it is generally not possible to determine the hue of a graphical object without also determining the saturation of that object. Again, BOZ uses separable graphical languages only.

determining the horizontal position of a graphical object, comparing two or more horizontal positions, and finding the midpoint of an interval defined by two horizontal positions. Table 3 shows the set of perceptual operators (POPs) admitted by two example primitive graphical languages: Horizontal Position and Shading.[4] It is interesting to compare the perceptual operators associated with each primitive graphical language. This exercise helps make explicit the task-specific usefulness of each graphical encoding technique. Note the difference in the number of computation operators supported by the Horizontal Position and Shading primitive graphical languages. For instance, human users can easily determine the difference between two horizontal positions but are not generally able to determine the difference between two shades.

Table 3: Perceptual Operators

Horizontal Position	**Shading**
(search operators)	*(search operators)*
determine-horz-pos	determine-shade
search-object-at-horz-pos	search-object-with-shade
search-any-horz-pos-object	search-object-and-shade
verify-object-at-horz-pos	verify-object-and-shade
(computation operators)	*(computation operators)*
horz-coincidence?	darker?
left-of?	lighter?
right-of?	same-shade?
horz-forward-projection	
horz-backward-projection	
determine-horz-distance	
find-horz-midpoint	

Perceptual operators are formalized using the same notation used for logical operators. For example, the `search-object-with-shade` search operator and `determine-horz-distance` computation operator are defined as follows:

[4]The perceptual operator sets associated with each primitive graphical language are not intended to be definitive or complete as we can easily think of many additional operations that could be performed using graphically expressed information. Rather, BOZ's current set of perceptual operators collects together a subset of possible perceptual inferences powerful enough to describe interesting graphic presentation-based problem solving.

```
(NLAMBDA search-object-with-shade (<OBJECT> <shade>)
  (ASK (Shading <OBJECT> <shade>)))

(NLAMBDA determine-horz-distance (<horzpos1> <horzpos2> <DISTANCE>)
  (DIFFERENCE <horzpos1> <horzpos2> <DISTANCE>))
```

The `search-object-with-shade` operator searches for an object in a graphic presentation having a shade equal to the value of `<shade>` and instantiates the variable `<OBJECT>` with the result. The `determine-horz-distance` operator determines the distance between two objects located at `<horzpos1>` and `<horzpos2>` in a graphic presentation and instantiates the variable `<DISTANCE>` with the result.[5]

3.2.2 Substituting Operators

The formal characterization of logical and perceptual operators using equivalent notations allows BOZ to design perceptual tasks that allow users to accomplish the same results as a logical task submitted to BOZ as input. The ultimate goal of BOZ is to arrive at that perceptual task that is equivalent to the original logical task, and that is most easily and efficiently performed by human users. To design a perceptual task, BOZ considers each logical operator (LOP) in a task description and searches the catalog of perceptual operators attempting to locate those POPs that can serve as substitutions for each LOP in the task. Perceptual operators can qualify as substitutions for logical operators in two ways. *Simple substitutions* are those in which a single perceptual operator can be shown to produce the same result as a logical operator. *Complex substitutions* are those in which two or more perceptual operators can be packaged together using a rule for the composition of operators to arrive at a complex perceptual operator that matches the logical operator.

Simple substitutions. A single perceptual operator qualifies as a substitution for a logical operator if and only if the POP can be shown to be a mathematical renaming of the LOP. The following definition formalizes the notion of simple substitution.

[5]Recall that upper-case arguments represent *uninstantiated* variables whereas lower-case arguments represent *instantiated* variables.

Definition 1 (simple substitution): *Let LOP_{search} and POP_{search} be search operators defined over the sets O, A, and PGL; where O is a set of objects, A a domain set of attributes, and PGL the discriminable values along some perceptual dimension:*

(NLAMBDA LOP_{search} (o, a)
(P o a))

(NLAMBDA POP_{search} (o', pgl)
(P' o' pgl))

substitution(POP_{search}, LOP_{search}) is true if and only if the following two conditions hold:

(i) there exists an injection: $A \Rightarrow PGL$,

(ii) the instantiations of the arguments o and o', and a and pgl match.

Let LOP_{comp} and POP_{comp} be computation operators defined over the domain set D (or union of domain sets $D = D1 \cup D2 \ldots \cup DN$) and the discriminable values along some perceptual dimension, PGL as follows:

(NLAMBDA LOP_{comp} (d1, d2, RESULT)
(F d1, d2, R))

(NLAMBDA POP_{comp} (pgl1, pgl2, RESULT)
(F' pgl1, pgl2, RESULT))

substitution(POP_{comp}, LOP_{comp}) is true if:

(i) there exists an injection: $D \Rightarrow PGL$,

(ii) $F = F'$.

For example, the substitution relation holds between the following two search operators since the argument instantiations match and we can assign a unique perceptual value (a label in this case) to each value in the domain set ORIGIN.

```
(NLAMBDA findFlightWithOrigin (<FLIGHT> <origin>)
  (ASK (Origin <FLIGHT> <origin>)))

(NLAMBDA search-object-with-label (<OBJECT> <label>)
  (ASK (Label <OBJECT> <label>)))
```

Note that the substitution relation does not hold between the `findFlightWithOrigin` and `search-object-with-shade` operators since the number of different origins exceeds the number of available shades (see Table 2), hence, violating condition (i).

The substitution relation holds between the following two computation operators since they both compute the same function (i.e., `DIFFERENCE`), and we can construct a mapping between the domain sets `DEPARTURE` and `ARRIVAL` and the set of unique horizontal positions.

```
(NLAMBDA computeLayover (<departure> <arrival> <LAYOVER>)
  (DIFFERENCE <departure> <arrival> <LAYOVER>))

(NLAMBDA determine-horz-distance (<horzpos1> <horzpos2> <DISTANCE>)
  (DIFFERENCE <horzpos1> <horzpos2> <DISTANCE>))
```

BOZ contains an operator substitution algorithm that considers each logical operator in a task description and locates all perceptual operators that qualify as substitutions for the logical operator as per Definition 1. Since there is only a small number of unique functions that the POPs in the catalog compute, BOZ categorizes the POPs in the catalog in advance based on the function they compute. Each class of POPs contains a schema that describes the function computed by the POPs in the class. The process of locating substitutions for each LOP is reduced to determining the function computed by a LOP and identifying the perceptual operator class that matches that function. Each POP in the class by definition qualifies as a substitution for the LOP. The following definition and theorem formalize the notion of an *operator equivalence class*.

> ***Definition 2 (operator equivalence classes):*** *An operator equivalence class, C, is a set of operators (op_1, ..., op_n) such that the relation substitution(op_i, op_j) holds for all $i, j \geq 1$ and $\leq n$. An operator equivalence class schema, S, is an arbitrary member of C.*

***Theorem 1**: Let C be an operator equivalence class and S_{class} the schema for C. If substitution(S_{class}, LOP) is true then substitution(POP_i, LOP) is true for all $POP_i \in C$.*

***Proof**: Immediately from Definitions 1 and 2.*

Table 4 lists twelve operator equivalence classes used to categorize the perceptual operators in the POP catalog.

Table 4: Equivalence Classes for Perceptual Operators

SEARCH OPERATORS

search: *Description*: Search a list of facts for an object with a specified attribute value. *Schema*: `(ASK (Predicate OBJ value)).`

lookup: *Description*: Search a list of facts about a particular object and report the value of a specified attribute of that object. *Schema*: `(ASK (Predicate obj VALUE)).`

search and lookup: *Description*: Search a list of facts for any object and report the value of a specified attribute of that object. *Schema*: `(ASK (Predicate OBJ VALUE)).`

***verify**: Description*: Search for a fact stating that a specified object has a specified attribute value. *Schema*: `(ASK (Predicate obj value)).`

COMPUTATION OPERATORS

equal: *Description*: Is one relational value equal to another? *Schema*: `(EQUAL x y EQ).`

lessthan: *Description*: Is one ordinal or quantitative value less than another? *Schema*: `(LESSP x y L).`

greaterthan: *Description*: Is one ordinal or quantitative value greater than another? *Schema*: `(GREATERP x y G).`

plus: *Description*: Computes the sum of two numbers. *Schema*: `(PLUS x y SUM).`

difference: *Description*: Computes the difference of two numbers. *Schema*: `(DIFFERENCE x y DIFF).`

times: *Description*: Computes the product of two numbers. *Schema*: `(TIMES x y PR).`

quotient: *Description*: Computes the quotient of two numbers. *Schema*: `(QUOTIENT x y Q).`

Each operator equivalence class contains perceptual operators drawn from the perceptual operator sets associated with each primitive graphical language. Table 5 shows the perceptual operators associated with the **search** and **subtraction** equivalence classes. These two classes group together all of the possible ways of searching for an item having a particular attribute (search), and determining the difference between two attribute values within the context of a graphic presentation.

Table 5: Members of Two Operator Equivalence Classes

search	**subtraction**
search-object-at-horz-pos	determine-horz-distance
search-object-at-vert-pos	determine-vert-distance
search-object-with-height	determine-height-difference
search-object-with-width	determine-width-difference
search-line-with-length	determine-difference-in-line-length
search-object-with-area	determine-area-difference
search-connected-object	subtract-labels
search-object-with-shading	subtract-table-entries
search-object-with-color	
search-object-with-label	
search-connected-object	
search-line-with-thickness	
search-line-with-dashing	
search-object-with-shape	
search-table-entry	

Complex substitutions. The substitution component sometimes exhausts the list of operator equivalence classes without successfully categorizing a LOP. One situation in which this occurs is when a logical search operator contains a relation that takes more arguments than the relations contained in any single perceptual search operator. For example, suppose an operator queries a 3-place relation such as: "find the brother of Heather," or "find the sister of Alison." This task can be represented using the following single LOP:

```
(LAMBDA find-certain-relative-of-x? (<person> <PERSON> <relation>)
  (ASK (Related <person> <PERSON> <relation>)))
```

If we consider the schemas for each of the equivalence classes in isolation we note that none of them formally qualifies as a simple substitution of `find-certain-relative-of-x`. When impasses of this sort occur, BOZ attempts to match the LOP to complex equivalence class schemas constructed from the set of simple class schemas using a composition of operators rule. The following definition formalizes the notion of operator composition:

Definition 3 (composition of search operators): *Let S and S' be operator equivalence class schemas:*

```
(NLAMBDA S (a1, a2, ... an)
 (<meta-command> (<predicate> a1 a2 ... an)))
```

and

```
(NLAMBDA S' (b1, b2, ..., bm)
 (<meta-command> (<predicate> b1 b2 ... bm)))
```

The composition of S and S', S ° S', is defined by the schema:

```
(NLAMBDA S-composition-S' (a1, ... (<predicate> b1 b2 ...
                                              bm), ... am)
 (<meta-command> (<predicate> a1 (<predicate> b1 b2 ... bm)
                                                ... an))
```

for some argument, a_i.

Hence, a *composition* of two search operators is a single search operator that uses another search operator as one or more of its arguments.

The following theorem establishes equivalence between LOPs and complex perceptual operator schemas defined through composition.

Theorem 2: *Let S" = S ° S' such that S and S' are schemas for operator equivalence classes, C and C'. If substitution(S", LOP) is true then substitution(POP_i ° POP_j, LOP) is true for all $POP_i \in C$ and all $POP_j \in C'$.*

Proof: *Immediately from Definitions 1 and 2.*

The following shows how a verify operator (`connected?`) can be composed with a lookup operator (`read-label`) to form a complex substitution for the `find-certain-relative-of-X` LOP.

```
(NLAMBDA read-label (<object>)
 (ASK (Label (find-connectee <object> <OBJECT>) <LABEL>)))
```

A final result is needed to show equivalence between logical procedures and perceptual procedures derived through operator substitution. This property insures that any perceptual procedure prescribed by BOZ will allow the user to obtain the same results as the corresponding logical procedure provided the perceptual procedure is performed correctly.

Theorem 3 (procedural equivalence): *Let LP be a logical procedure inductively defined as follows:*

$LP \Rightarrow LOP$
$LP \Rightarrow LP \bullet LP$
$LP \Rightarrow$ ***if*** LP ***then*** LP
$LP \Rightarrow LP^m$
$LP \Rightarrow$ ***repeat*** LP ***until*** LP
$LP \Rightarrow e$

Any perceptual procedure, VP, derived from LP through operator substitution is equivalent to LP.

Proof *(by induction): By Definition 1, a logical procedure consisting of a single LOP is equivalent to any perceptual procedure consisting of a single POP if and only if substitution(POP, LOP) is true. If substitution(POP_1, LOP_1) and substitution(POP_2, LOP_2) are true then $LOP_1 \bullet LOP_2 = POP_1 \bullet POP_2$ since, by definition, the output of any operator is uniquely determined by the input and the function computed by the operator. Since logical implication is context-free, (if LOP_1* ***then*** *LOP_2) is equal to (if POP_1* ***then*** *POP_2) if and only if substitution(POP_1, LOP_1) and substitution(POP_2, LOP_2) are true. Since iteration is defined as $LOP_i^m = (LOP_{i1} \bullet LOP_{i2} \bullet \ldots \bullet LOP_{im})$, $LOP_i^m = POP_i^m$ when substitution(POP_i, LOP_i) is true.*

Figure 14 shows the classifications for the logical operators in the airline reservation task. Each operator in the task matches the schema of a single equivalence class. For each logical operator in the task, a set of perceptual operators initially qualify as substitutions, namely those perceptual operators that are members of the operator equivalence classes named in parentheses in Figure 14. For example, the set of perceptual operators associated with the **search** equivalence class are proposed as substitutions for the `findFlightWithOrigin` and `findSeat` operators. Similarly, the operators of the **subtraction** class match the description of the `computeLayover` operator.

```
findFlightWithOrigin (search)
findDestination (lookup)
available? (lookup)
determineDeparture (lookup)
determineArrival (lookup)
computeLayover (subtraction)
layoverLessThanX? (lessthan)
determineCost (lookup)
addCosts (addition)
costLessThanX? (lessthan)
findSeat (search)
emptySeat (lookup)
```

Figure 14: Operator Classifications for the Airline Reservation Task

It is important to note that BOZ has not yet decided *which* perceptual operator to choose in each case. Decisions about which perceptual operators to match with each logical operator are subject to further constraints computed by the perceptual data structuring and perceptual operator selection components described in Sections 3.3 and 3.4. Consequently, what BOZ has produced at this stage is a space of perceptual procedures that may be selected according to additional design criteria.

3.3 Perceptual Data Structuring

Perceptual data structuring is the technique used to implement the second type of cognitive advantage of graphic presentations: that graphic presentations sometimes allow users to spend less time searching for needed information. The perceptual data structuring component examines the information required to perform each logical operator in a task.

Two types of analyses are performed using this information. First, by noting the domain sets that each logical operator is defined over, BOZ determines what information should appear in a graphic designed to support a task. Second, by analyzing the relationships between the operators in terms of the domain sets of information they manipulate, it is determined: (a) how information shared by several operators should be collected in the same spatial locality and graphically encoded using the same primitive graphical language; (b) and how information not shared among operators can be partitioned into distinct presentations. The perceptual data structuring component produces a *perceptual data structure specification* that outlines the presentations that will be used to support the task, the information that should appear in each presentation, and how information is to be grouped into graphical objects within each presentation. The perceptual data structuring component *does not* decide which primitive graphical languages are to be used to encode each type of information in the presentations. These decisions are made by the perceptual operator selection component (Section 3.4).

The remainder of this section describes a scheme that analyzes relationships between operators by representing each operator as a vector defined over a collection of domain sets. Relationships between vectors are determined by identifying common domain sets occurring in vectors. A complete sketch of all relationships between vectors reveals how information is to be collected together into graphical objects and partitioned among graphic presentations.

3.3.1 Operator Vectors

Recall that every task description is defined over a finite collection of domain sets. When taken together, all of the domain sets used by a task description form a feature space. A *feature space* is formally defined as the cross product of all domain sets spanned by a task description. Figure 15 shows an example of a feature space defined over the domain sets that pertain to the airline reservation task.

[flight] x [origin] x [destination] x [departure] x [arrival] x [layover] x [cost] x [seat] x [availability]

Figure 15: Feature Space for the Airline Reservation Task

Each logical operator in a task description computes a relation of the form (p, o, a_i, ... a_j) or (f, a_i, ... a_j) for some i and j less than or equal to the total number of domain sets drawn from a feature space, called a *vector*. Vectors of the first form are called *search vectors*. Vectors of the second form are called *computation vectors*. The first element in a search vector, p, names the *predicate* that the search vector computes. The second element in a search vector, o, is the *object*, on which p is predicated. The remaining elements of a search vector, a_i, ... a_j, are *attribute values* for the predicate, p. The first element in a computation vector, f, names the *function* that the operator computes. The remaining elements in a computation vector, a_i, ... a_j, are the *arguments* to the function, f.

The operators in the airline reservation task define the vectors shown in Figure 16. The vectors in Figure 16 indicate the relationships between the domain sets shown in Figure 14 as defined by the airline reservation task.

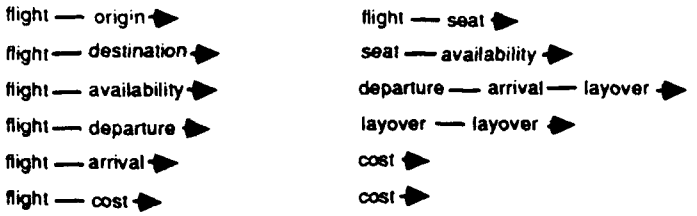


Figure 16: Vectors for the Airline Reservation Task.

3.3.2 Relationships Between Vectors

Relationships between vectors are determined in the following way. Search vectors sv_1 and sv_2 are said to be *conjoint* when they contain common objects, o. Search vectors sv_1 = (p, o, a_1, ... a_i) and sv_2 = (p, o, b_1, ... b_j) are *parallel* when there exists a computation vector cv = (f, c_1, ... c_k) such that there exists some c_m in a_1, ... a_i and some c_n in b_1, ... b_j for m and n in (1 ... k). Search vectors sv_1 and sv_2 are said to be *orthogonal* if some

attribute, a_i, of sv_2 appears as the object, o, in sv_1. Search vectors sv_1 and sv_2 are orthogonal to each other when some attribute, a_i, of sv_2 appears as the object, o, in sv_1, and some attribute, a_i, of sv_1 appears as the object, o, in sv_2. Search vectors sv_1 and sv_2 are *disjoint* when they are neither parallel or orthogonal.

Relationships between vectors determine the perceptual structuring of information in a presentation in the following way. *Conjoint vectors* group together attributes that pertain to the same object. Consequently, these attributes should be encoded in a single graphical object in order to minimize eye movement over the graphic when searching for that object and its attributes. *Parallel vectors* indicate that some perceptual operator(s) requires that two or more different objects and their attributes be coordinated by the user in order to draw a particular inference. Consequently, both objects should appear in the same presentation and be encoded using the same primitive graphical language. *Disjoint vectors* indicate that no perceptual operator requires that two or more objects be coordinated to drawn an inference. Vectors of disjoint operators can be supported in different presentations since the information they manipulate is never used together. *Orthogonal vectors* indicate part-of relationships between objects. Information manipulated by orthogonal vectors is presented in separate nested presentations. That is, the user should be able to view the part-of presentation by making an appropriate selection in the first presentation.

Figure 17 shows the relationships between the vectors of Figure 16 for the airline reservation task.

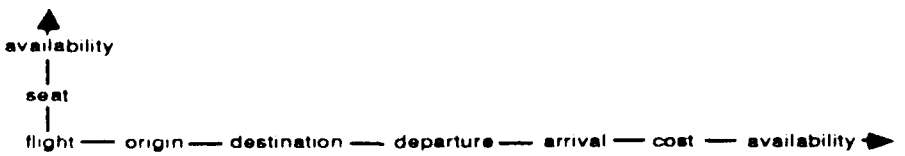


Figure 17: Vector Relationships for the Airline Reservation Task

Since the vectors pertaining to the origin, destination, departure, arrival, cost, and availability of a flight are conjoint, they should be encoded in the same graphical object. The vectors pertaining to seats and flights are orthogonal indicating that each seat object is a part of a flight object. ~~ seating information should be presented in a separate graphic presentation that is nested inside each flight box.

Figure 18 shows the initial perceptual data structure specification for the airline reservation task.

```
(NESTED (PRESENTATION1 (flight (Origin Destination Departure
                                Arrival Cost Availability))
        (PRESENTATION2 (seat (Seat Availability))))
```

Figure 18: Initial Perceptual Data Structure Specification for the Airline Reservation Task

This initial data structure indicates how data will be grouped together into complex graphical objects and distributed among distinct graphic presentations. It is important to note that it is not yet been decided how facts about the origin, destination, departure, etc. of a flight are to be graphically encoded in the graphic. That is, BOZ has not yet associated the names of primitive graphical languages with the predicate names appearing in the perceptual data structure specification. Which primitive graphical languages to associate with each predicate is determined by the perceptual operators selected to substitute the logical operators in the task. Note that information about flight numbers and layovers will not be encoded in any graphic. This has occurred since information about flight numbers is never used in the task, and facts about layovers are produced as the results of a perceptual operator, `computeLayover`.

3.4 Perceptual Operator Selection

The perceptual operator selection component chooses a single perceptual operator to substitute each logical operator from the list of possibilities generated by the perceptual operator substitution component. Selecting a single perceptual operator to substitute each

logical operator accomplishes two things: (a) collapses the space of possible perceptual procedures to a single perceptual procedure judged to be the easiest to perform; and (b) allows BOZ to design a single accompanying graphic that supports human performance of the selected perceptual procedure.

Three important issues constrain the selection of perceptual operators. First, since the goal is to arrive at a perceptual procedure that minimizes the effort required to correctly complete a task, for each logical operator we wish to choose that perceptual operator that is performed most efficiently and accurately by human users. A first criteria for operator selection involves estimating the relative performance efficiency and accuracy of the perceptual operators. Second, recall that each perceptual operator is associated with a primitive graphical language that must be used to graphically encode information manipulated by that operator. A second criteria when selecting operators is that the representational power of the primitive graphical language associated with a candidate perceptual operator be sufficient to encode the logical facts manipulated by the operator. Third, recall that the perceptual data structure specification produced by the perceptual data structuring component constrains some domain sets of information to be represented using a single graphical object, or using the same primitive graphical language. A third criteria for operator selection is that the primitive graphical languages associated with the selected perceptual operators be combinable such that they result in coherent graphical representations that agree with the perceptual data structure specification for the task. The following sections explain how each of the operator selection criteria are used by BOZ to ultimately arrive at a single perceptual procedure and graphic presentation.

3.4.1 Human Performance Rankings for Perceptual Operators

The most important criteria when selecting a perceptual operator is choosing that operator that allows the human user to obtain the results of the operator most efficiently and accurately. To determine which of a set of perceptual operators is likely to be the most

performance effective, BOZ uses a two-tier ranking system that is a generalization of the approach used in Mackinlay's APT program. The first tier ranks the equivalence classes for operators appearing in Table 6 in order of their relative difficulty. For instance, **search** operators require more effort to perform than **lookup** operators. Consequently, they are always awarded the most efficient perceptual operators. The second tier ranks the perceptual operators within each operator class. For instance, determining the horizontal distance between two points on a scale is generally performed more efficiently than determining the difference between two sloped lines. The rankings were generated using a combination of two methods: (a) theoretical predictions based on a more fine-grained consideration of each perceptual operator [Card, Moran, and Newell, 1983; Ullman, 1984]; and (b) experimental observations of human perceptual task performance that incorporate measures of difficulty, accuracy, and perceptual salience [Cleveland and McGill, 1983; Ullman, 1984; Jenks and Knos, 1961; Teghtsoonian, 1965; Davidoff, 1987]. Table 6 shows the rankings for perceptual operator equivalence classes, and the rankings for the perceptual operators in each class.

Table 6: Ranking of Perceptual Operators and Equivalence Classes

A. Class Rankings:

1. plus, difference, quotient, times
2. verify
3. search
4. max, min
5. lessthan, greaterthan
6. equal
7. search and lookup
8. lookup

B. Operator Rankings:

SEARCH OPERATORS

search, verify: {Visibility, HorzPos, VertPos, Shape, Connectivity, Shading, Height, Width, LineDashing, LineLength, LineThickness, Labels, Area}

lookup, search and lookup: {Shading, Shape, Labels, Height, Width, LineDashing, LineThickness, Connectivity, HorzPos, VertPos, LineLength, Area, Visibility}

COMPUTATION OPERATORS

equal: {Labels, Shading, HorzPos, VertPos, Shape, LineDashing, Height, Width, LineThickness, LineLength, Connectivity, Visibility, Area}

lessthan, greaterthan: {Shading, HorzPos, VertPos, Height, Width, LineThickness, LineLength, Labels, Connectivity, Shape, LineDashing, Visibility, Area}

plus, times: {Height, Width, LineLength, LineThickness, HorzPos, VertPos, Labels, Connectivity, Shading, LineDashing, Shape, Area}

difference, quotient: {HorzPos, VertPos, Height, Width, LineLength, LineThickness, Labels, Connectivity, Area, Shading, Shape}

3.4.2 Primitive Graphical Language Expressiveness

The second criterion used during operator selection is that a selected perceptual operator must be associated with a primitive graphical language that is powerful enough to encode the logical facts manipulated by that operator. For example, even though the `search-shaded-object` is the most efficiently performed search operator, it cannot be selected to substitute the `findFlightWithOrigin` logical operator since the number of different cities exceeds the number of different shades. When a selected perceptual operator fails to meet the expressiveness needs of a logical operator it is disqualified and the next highest ranking perceptual operator is considered. The interested reader can consult Mackinlay (1986) for a thorough analysis of primitive graphical language expressiveness. BOZ, like all other recent presentation systems, adopts Mackinlay's mechanism for deciding expressiveness.

3.4.3 Operator Combinability

The third criterion for operator selection concerns the combinability of perceptual operators. Suppose we have selected the `stack-heights` perceptual operator to substitute the `addCosts` logical operator in the airline reservation task and are currently selecting a perceptual operator to substitute the `findFlightWithOrigin` operator. Suppose that we are currently considering the `determine-slope` perceptual operator as a candidate selection. Recall that the perceptual data structuring component has indicated that the information relevant to these two operators should be encoded in the same graphical object. Every perceptual operator has associated with it a *graphical presentation object* that is used to graphically encode the information manipulated by that object. For example, the graphic presentation object for the `stack-heights` and `determine-slope` operators are <rectangle> and <line>, respectively. Note that the information relevant to the two operators cannot be encoded in the same graphical object. That is, it is meaningless to speak of the slope of a rectangle or the height of a line. Consequently, these two operators are not combinable and we must disqualify `determine-slope` as a candidate for

selection. Now suppose we move on and consider the `read-label` perceptual operator. Note that the two operators are indeed combinable. Even though the graphic presentation object for the `read-label` operator is <label> and the graphic presentation object for the `stack-heights` operator is <rectangle>, the two graphical objects can be combined to form a labeled rectangle.

The next two sections describe how the set of graphic presentation objects and a set of graphic object composition rules are used by BOZ to decide combinability of perceptual operators.

3.4.3.1 Graphic Presentation Objects

Each primitive graphical language has a *graphic presentation object* associated with it, either: <point>, <line>, <rectangle>, <polygon>, or <label>. The graphic presentation objects for a primitive graphical language are those graphic objects that support the performance of the perceptual operators that are associated with that graphical language. For example, the graphic presentation object for the Height primitive graphical language is <rectangle>. Note that only this object makes the perceptual operators associated with the Height language meaningful. That is, it would be impossible to determine the height of a point since a point by definition has no spatial extent.

Table 7 lists the graphic presentation objects associated with each of the primitive graphical languages.

Table 7: Graphic Presentation Objects of the Primitive Graphical Languages

Horizontal Position = <point>
Vertical Position = <point>
Height = <rectangle>
Width = <rectangle>
Line Length = <line>
Area = <polygon>
Connectivity = <line>, <point>
Shading = <polygon>, <rectangle>
Labels = <label>
Color = <point>, <line>, <rectangle>, <polygon>
Line Thickness = <line>, <rectangle>, <polygon>
Line Dashing = <line>, <rectangle>, <polygon>
Shape = <polygon>
Visibility = <point>, <line>, <rectangle>, <polygon>, <label>

The first step in deciding perceptual operator combinability is to determine the graphic presentation object of a candidate perceptual operator.

3.4.3.2 Composition Rules for Graphic Presentation Objects

The second step in deciding operator combinability is to compare the graphic presentation object of the perceptual operator currently being considered with the presentation objects of all previously selected operators that appear in the same vector in the perceptual data structure specification. If the graphic presentation object matches those of the previously chosen operators then the new operator is combinable. If the presentation object does not match, BOZ attempts to demonstrate that they are combinable using the set of composition rules for graphic objects given in Table 8.

Table 8: Composition Rules for Graphic Presentation Objects

Mark Composition Rules:
RULE 1: <point> + <point> = <point>
RULE 2: <point> + <line> = <line>
RULE 3: <line> + <line> = <line>
RULE 4: <rectangle> + <point> = <rectangle>
RULE 5: <rectangle> + <rectangle> = <rectangle>
RULE 6: <polygon> + <point> = <polygon>
RULE 7: <polygon> + <polygon> = <polygon>
RULE 8: <label> + <label> = <label>
RULE 9: <label> + <line> = <line>
RULE 10: <label> + <rectangle> = <rectangle>
RULE 11: <label> + <polygon> = <polygon>

Axis Composition Rules:
RULE 12: <horz-axis> + <horz-axis> = <horz-axis>
RULE 13: <vert-axis> + <vert-axis> = <vert-axis>
RULE 14: <horz-axis> + <vert-axis> = <cart-axis>

Network Composition Rules:
RULE 15: <node-link-node> + <node-link-node> = <node-link-node>

Each composition rule describes how individual presentation objects can be legally composed to form a single presentation object that inherits all of the graphical properties of the constituent objects. For any new perceptual operator and set of previously selected operators, the new operator is combinable if and only if a rule can be found that maps the complete set of graphic presentation objects into another legal presentation object.

Applying the operator selection strategy to the set of possible perceptual operators obtained for the airline reservation procedure yields the perceptual procedure shown in Figure 19. For each logical operator, BOZ has selected the most efficient available perceptual operator as a substitute within the expressiveness and combinability constraints described above. It is interesting to note that the logical operator `findFlightWithOrigin` has been substituted by the `search-object-with-label`, a very low-ranking perceptual operator. This example illustrates how the expressiveness and combinability constrains operator selection. A more appealing substitution for `findFlightWithOrigin` would have been a perceptual operator in which the user locates a flight with a particular origin by searching a horizontal scale or by searching for an object having a particular shape. The reason that these two more attractive operators were not selected is because the higher-ranking `computeLayover` had already staked claim to the Horizontal Position operators, and the Shape primitive graphical language unfortunately does not offer enough unique shapes to represent all ten different possible cities of origin. Consequently, due to these

constraints imposed by the other competing operators, the findFlightWithOrigin operator was relegated to the more difficult perceptual task of searching for a labelled item.

```
(TASK airlineReservation
   (while (search-object-with-label FLIGHT 'pit) do
    (if (shaded? flight) then
     (read-label flight LAYOVERCITY)
     (determine-horz-pos flight ARRIVAL)
     (while (search-object-with-label CONNECTING layovercity) do
        (if (shaded? connecting) then
         (read-label flight FINALDESTINATION)
         (if (same-labels? finaldestination 'mex) then
          (determine-horz-pos connecting DEPARTURE)
          (determine-horz-distance departure arrival LAYOVER)
          (if (and (right-of? departure arrival)
                   (left-of? layover)) then
           (determine-height flight COST1)
           (determine-height connecting COST2)
           (stack-heights cost1 cost2 TOTAL)
           (if (shorter? total) then
            (repeat
             (search-object-with-label flight SEAT1)
            (until (shaded? seat1)))
            (read-label seat1 SEATNUM1)
            (repeat
             (search-object-with-label connecting SEAT2)
            (until (shaded? seat2)
            (read-label seat2 SEATNUM2)]
```

Figure 19: The Perceptual Airline Reservation Procedure

Figure 20 shows the final perceptual data structure specification for the airline reservation task after the composition rules have been applied.

```
(NESTED (PRESENTATION1 (flight ((Origin Labels) (Destination Labels)
                         (Departure HorzPos) (Arrival HorzPos)
                         (Cost Height) (Availability Shading))
                        <rectangle>)
      (PRESENTATION2 (seat ((Availability Shading)) <rectangle>)))
```

Figure 20: Final Perceptual Data Structure Specification for the Airline Reservation Task

Note that each predicate appearing in the perceptual data structure specification has been associated with a single primitive graphical language. In each case the primitive graphical language chosen is precisely that language associated with the perceptual operators that have been selected to manipulate that type of information.

3.5 Graphic Presentation Rendering

The rendering component uses the perceptual data structure specification to translate logical facts submitted with the logical procedure to graphical facts. *Graphical facts* use the primitive graphical languages to equivalently express the original set of logical facts in a graphical format. Graphical facts are rendered using a graphic design that agrees precisely with the perceptual data structure specification designed by BOZ for the task. The rendering component produces a fully rendered graphic presentation of the graphical facts.

3.5.1 Translating Logical Facts to Structured Graphical Facts

A prerequisite to graphically rendering arbitrary sets of logical facts on the computer screen is a notation for representing graphical facts that is equivalent to the notation used to express logical facts. To accomplish this, Mackinlay's formalism for expressing graphical facts is used, and this has been shown to be equivalent to a logical representation of the same relational information [Mackinlay, 1986]. Mackinlay's formulation allows logical facts to be expressed using each of the primitive graphical languages given in Table 2. Graphical facts expressed in a primitive graphical language take the following form: `(PGL <object> <value>)`. For example, the facts in Figure 21 describe a square-shaped graphic object that is shaded black and positioned along a horizontal axis.

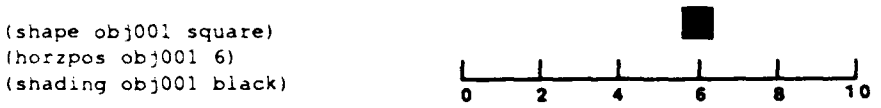


Figure 21: Example Graphical Facts

Figure 22 shows how logical facts (left side) about airline flights are renamed to graphical facts (right side) when the mapping given in the perceptual data structure specification (Figure 20) is applied.

```
(origin flight117 pit)                    (label flight117 pit)
(origin flight239 hou)                    (label flight239 hou)
(destination flight117 hou)               (label flight117 hou)
(destination flight239 mex)               (label flight239 mex)
(departure flight117 10:00)              (horzpos flight117 10)
(departure flight239 15:00)              (horzpos flight239 15)
(arrival flight117 12:50)              (horzpos flight117 12.83)
(arrival flight239 17:15)              (horzpos flight239 17.25)
(cost flight117 179)                     (height flight117 1.79)
(cost flight239 239)                     (height flight239 2.39)
(availability flight117 ok)        (shading flight117 whiteshade)
(availability flight239 ok)        (shading flight239 whiteshade)
```

Figure 22: Translated Airline Reservation Facts

A final notation is needed for expressing collections of graphical facts whose structure agrees with the perceptual data structure specification for the task. A *structured graphical fact* corresponds to the "gestalt wholes" defined by the perceptual data structure specification. For example, the facts in Figure 23 show how the graphical facts (Figure 21) are structured according to the perceptual data structure specification for the airline task given in Figure 20.

```
(((labels flight117 pittsburgh)
  (labels flight117 hou)
  (horzpos flight117 10)
  (horzpos flight117 12.83)
  (height flight117 1.79)
  (shading flight117 whiteshade))
((labels flight239 hou)
  (labels flight239 mex)
  (horzpos flight239 15.00)
  (horzpos flight239 17.25)
  (height flight239 2.39)
  (shading flight239 whiteshade)))
```

Figure 23: Structured Graphical Facts

3.5.2 Rendering Graphical Facts

The rendering component automatically displays arbitrary sets of structured graphical facts on the computer screen. This is accomplished by considered each structured fact, determining the form in which it is to be presented by consulting the perceptual data structure specification, and rendering the image of the fact on the screen. The rendering algorithm uses an object-oriented approach to rendering graphic presentation objects and

their graphical properties. For every type of graphic presentation object (i.e., <point>, <line>, <rectangle>, <polygon>, and <label>) there exists a corresponding *display object* that can be rendered on the screen. To expedite the rendering of display objects, drawing primitives are used to create images rather than bitmap displays. Display objects can inherit one or more of a set of *display methods* that render the graphical properties of a display object. Display methods are defined for each of the primitive graphical languages in Table 2. For presentations that do not use horizontal and vertical position to encode information, a simple displacement scheme is used to minimize occlusion of display objects by other display objects. Scales and guidelines are automatically computed, drawn, and labeled using the DOMAINSETS field in the logical procedure description. Fonts have been chosen arbitrarily and standardized. Nested graphics are implemented by mouse-sensitive buttons that are always placed in the lower left corner in rectangles and polygons, and immediately on top of points and lines. Customized methods are automatically attached to the buttons that cause the nested graphic to be rendered when the button is selected.

Figure 24 shows a fully rendered set of graphical airline facts. As specified by the perceptual data structure specification, the presentation consists of a single type of graphic object (i.e., a flight box) that inherits four graphical properties (i.e., horizontal position, shading, height, and labels). Selecting the SEATS button for any flight causes the nested seating chart for that flight to be rendered. A rendered seating chart is shown in Figure 25.

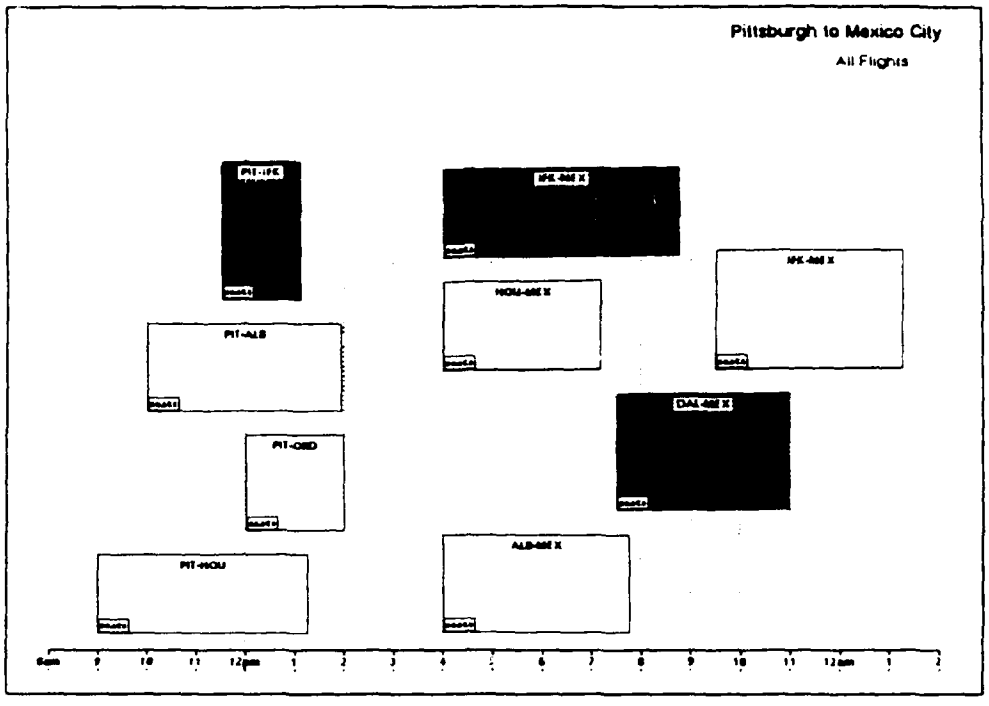


Figure 24: Rendered Graphic Airline Schedule.

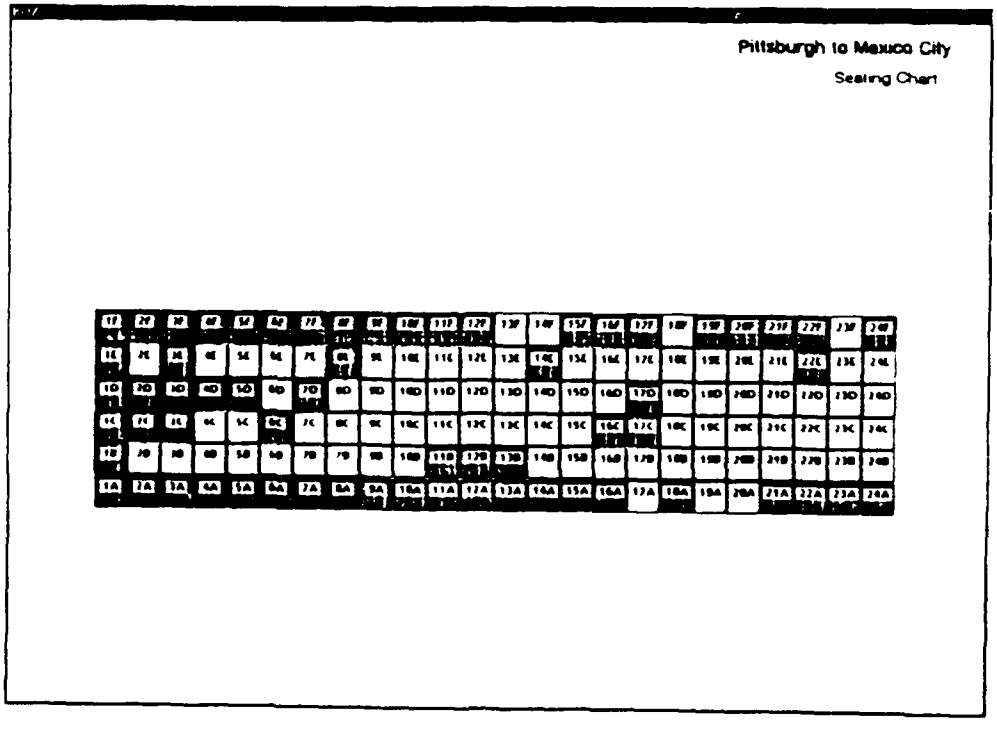


Figure 25: Rendered Seating Chart.

3.5.3 Interactive Graphic Presentation Objects

BOZ contains an additional package that allows the graphics generated by the rendering component to support two-way interactions between sets of logical facts and their graphic images. In addition to being able to effect changes in a graphic presentation through manipulations of the internally-stored logical facts, the graphic objects in the presentations can be manipulated by the user to effect changes in the set of stored logical facts. For example, the upper, left-most flight box in the presentation in Figure 24 indicates that there is flight from PIT to JFK leaving at 11:30am with no available seats costing $400. The user may simultaneously change the graphical and internal representation of this information by simply mouse-selecting the flight box and moving it to a new location, changing its shading, or increasing or decreasing its height.

3.6 Limitations of BOZ's Automated Design Approach

The following surveys important limitations of BOZ's automated approach to graphic presentation design.

3.6.1 Limitations of the Task Description Language

A limitation of BOZ's task description language is that using the present notation LOPs can only accept and return single values. A common situation in which it is desirable to return sets of values is in describing complex search strategies. Using the present notation, the default search space for any LOP is the entire factbase of relations. A simple task not describable under this condition is binary search in which the input to successive operators is a halving of the original factbase. This same problem is suffered by production system languages such as OPS5 [Forgy, 1979]. Working memory elements must be tagged to indicate that they are to be excluded from future search.

3.6.2 Limitations of the Perceptual Operator Substitution Component

An important limitation of BOZ's perceptual operator substitution component concerns the types of logical operators that BOZ can substitute. BOZ presently handles LOPs that contain queries over quantitative, nominal, and ordinal information only. Roth (1990) points out that quantitative, nominal, and ordinal information is a limited type of relational information that is derived through the notion of *set ordering*. Roth (1990) articulates a more complete space of relational information types in the context of graphic design. These information types are shown in Figure 26.

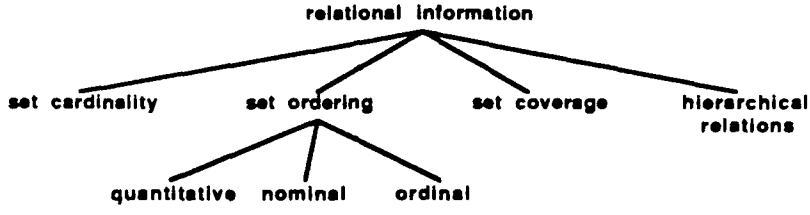


Figure 26: Roth's Relational Data Types

Roth shows that many popular graphic designs present information types other than those derived through set ordering. For example, the pictogram shown in Figure 27 depicts information about *set cardinality* and supports a simple counting task.

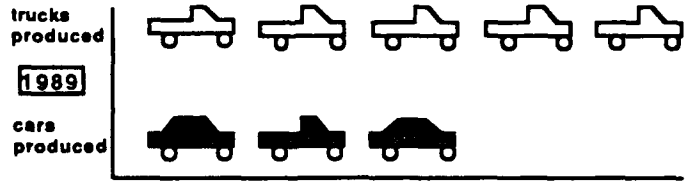


Figure 27: Pictograms: A Type of Graphic Not Designable by BOZ

BOZ's present perceptual operator catalog does not contain counting operators. Consequently, BOZ is unable to design presentations that support cardinality judgements such as the pictogram shown in Figure 27.

3.6.3 Limitations of the Automated Perceptual Operator Selection Component

There are many limitations of BOZ's automated perceptual operator selection strategy. First, it is important to note that there exists no general strategy that always chooses the most efficiently or accurately performed perceptual operator, including those based on experimental observations and detailed theoretical predictions. Chapter 4 discusses the many factors influencing perceptual task performance, each of which have been shown to introduce variance strong enough to overturn the results of any particular experiment, making strong generalizations of theoretical and experimental results inappropriate. What we can hope to achieve in an automated graphic design tool is a codified set of operational design principles that perform satisfactorily across interesting tasks and graphics. An important feature of BOZ is that the ranking scheme used for operator selection is parameterized and easily modified. Should future psychological data or the characteristics of any particular application or user population show BOZ's operator rankings to be inappropriate, the rankings can be quickly changed by simply reordering the entries in Table 6.

Second, many task domains make use of domain-specific graphic conventions for which practitioners of that domain have acquired practiced skill in using. It is not always the case that these conventions were chosen based on which conventions were the most cognitively efficient, but rather what informally seemed to comprise a useful notation at the time the graphic convention was designed. Without specific knowledge of a problem domain, an automated graphic design tool is unable to identify and select operators that correspond to existing graphic conventions. BOZ, like APT [Mackinlay, 1986], allows the designer to intervene and manually select perceptual operators in order to support existing conventions.

Third, when two logical operators are ranked evenly, the logical operator that appears first in the submitted task description is allowed to choose a perceptual operator first. This

occurs since BOZ contains no additional selection criteria to apply when ties between operators occur. Within the scope of BOZ's design criteria it is not clear that any tie-breaking criteria exist that have distinct advantages over the arbitrary ordering strategy used by BOZ.

Fourth, the rankings of perceptual operators used by BOZ's perceptual operator selection component embody the following assumption: the performance difficulty of any operator is constant over the entire range of input values. That is, BOZ assumes that it is never the case that one perceptual operator is performed more efficiently than a second operator given one set of input values, and performed less efficiently given a second set of inputs. This assumption is known to be in general inappropriate. For example, judging the distance between two points on scale that are aligned three units away from one another appears to be easier than if the points are aligned, say, thirty-nine units apart. This phenomenon occurs because some input data allow the user to exploit more low-level perceptual capabilities such as *subitizing* [Klahr, 1973]. Similarly, Ware and Beatty (1988) show that accuracy of color perception is not uniform over the space of perceivable colors. A more reliable perceptual operator ranking system might be achieved by making the set of rankings functionally dependent on the set of logical facts to be displayed. That is, rather than using a fixed set of operator rankings, a different set of rankings would be computed for each different set of logical facts. Since the data dependencies are likely to vary between primitive graphical languages (and perhaps between the individual operators themselves) is not clear what would constitute a reliable ranking function, what variables should be included in the function, and the practicality of executing such a function and still maintaining a run time that meets real-time graphic presentation standards.

Fifth, another data-dependent graphic design issue to which BOZ is currently insensitive is the number of data values that can be practically displayed using a particular design. For example, the graphic airline presentation is unlikely to be effective when the number of

flights to be displayed is more than fifteen or twenty due to unavoidable occlusion of some flight boxes by others. The maximum number of data values presentable using any presentation can be easily estimated by determining the smallest discriminable graphic object size and the spatial extent of the fovea. For example, the advantages of the graphic airline schedule are likely to disappear when the size of each flight box becomes too small or the width and height of the presentation become too large to perceive in two or three eye fixations. Note that it is trivial to extend BOZ to enforce this constraint by setting a practical limit on the size of the domain sets that can be manipulated by each perceptual operator, disqualifying any perceptual operator when the size of a domain set to be presented exceeds it limit.

Fifth, the first tier of the operator ranking scheme, that which rates the relative importance of each operator type is oversimplified in that it does not consider the number of times that each operator will be executed during performance of the procedure. Note that a difficult operator that is performed once during a procedure may be less important than a slightly less difficult operator that is performed many times. In the worst case this oversimplification can result in the most efficient perceptual operators being awarded to logical operators that comprise a comparatively small part of the user's task, while more frequently performed logical operators are relegated to the less efficient perceptual operators. This occurs since there is no way to discern in advance the number of times each logical operator in a procedure will be performed during execution of that procedure. For specific design applications, one strategy for circumventing this problem would be to run the procedure in advance on several inputs and count the number of times each operator is executed. This information could be used to hand optimize BOZ's operator selections. The procedure interpreter tool described in Chapter 4 can be used to accomplish this type of off-line analysis.

3.6.4 Limitations of the Automated Rendering Component

There are several important limitations of the automated rendering component. First, the rendering component is incapable of rendering presentations that make use of domain-specific conventions. For example, airline seating charts typically orient the aircraft pointing up, lower numbered seats appearing at the top and higher number seats at the bottom. Since BOZ's rendering component has no knowledge of this convention, the seats are arranged in increasing order from left to right as are the hours along the time scale.

Second, many presentations depict realistic information such as spatial arrangements and shapes that do not encode information vital to the task at hand but preserve many features of a real-world artifact in an artificial representation. For example, airline seating charts typically depict the aisle separating the two halves of the plane. Some seating charts also use chair-shaped icons to represent seats instead of the generic box-shape used in BOZ's presentation. BOZ of course has no knowledge of these conventions. Note that despite these two limitations it is still possible to locate any seat. What may be lost is a familiarity and practice that users may have already acquired using other conventions.

Third, there are a number of issues pertaining to how humans perceptually group information that have important implications for image rendering and to which BOZ's rendering component is not sensitive. These issue arise when objects in a graphic share similar properties. For example, when viewing a set of marks in a graphic, people tend to perceive marks positioned in close spatial proximity as a single perceptual unit. That is, people typically perceive "xx xx" as two objects and "x x x x" as four objects. Graphic objects that appear in a continuous line, or that have similar shapes or colors are also likely to be grouped together. The basic problem arising from these effects is that the user of a graphic may automatically and unavoidably draw inferences from a set of graphical facts that are not included as part of a prescribed perceptual task. Note that perceptual grouping effects are not always bad. For example, the Labels primitive graphical language makes

active use of perceptual grouping effects by placing any label associated with a graphic object immediately below that object. Due to the proximity of the label to the object, the user is immediately able to infer with which object the label is associated. Several techniques exists for diminishing the effects of perceptual grouping when the effects of grouping are negative. Chambers et al (1983) propose a "jittering" technique for breaking up perceptual groupings that arise from spurious horizontal or vertical alignments.

Fourth, a basic feature of human visual perception is that the perception of graphically encoded quantities is routinely erroneous. Consequently, for some primitive graphical languages, it is insufficient to simply map the values from a logical domain set onto the value of the perceptual domain set. Fortunately, many errors in perceived quantities behave systematically facilitating the use of "correction formulas" to align perceived and actual values. Weber's Law and Steven's Law are two techniques for adjusting continuous perceptual scales such that perceived distances between values along the scale agree with actual distance. Teghtsoonian (1965) shows how perceived area typically agrees with the actual area raised to a power of 0.8. Flannery (1956) proposed a formula for correcting area and diameter judgement for circle. BOZ applies these simple correction formulas to all graphical facts before they are rendered. Unfortunately, not all perceptual distortion phenomena can be accounted for at the level of simple information-extraction tasks. For example, Kosslyn (1985) argues that when judging goodness of fit of a line through a set of points in a graph, people tend to report a figure more consistent with r^2 than r. Distortion effects arising from more complex perceptual tasks seem to require their own customized correction formulas, disallowing the use of simple formulas that are applied generally to the primitive graphical languages.

Fifth, when using labels and table entries, BOZ arbitrarily chooses among typographic conventions such as font, font size, and case. It has been demonstrated that typography

can affect several parameters of human performance, most importantly search time [Vartabedian, 1971].

CHAPTER 4

THEORETICAL MEASURES OF GRAPHIC DESIGN EFFECTIVENESS

This chapter describes theoretical tools used to measure to what extent a BOZ-designed presentation and procedure exploit the six advantages of graphical presentations discussed in Chapter 1. To make these analyses more rigorous, BOZ contains a simulation component that allows the logical and perceptual procedures handled by BOZ to be executed. The simulation component tracks two important human task performance parameters during execution of a procedure: (1) the number of times each task operator is fired; and (2) the total number of items searched. These measures can be used to generate theoretical predictions about the effectiveness of a BOZ-designed presentation with respect to a task, and to compare the relative effectiveness of each of a set of alternative presentations. The ultimate goal of the automated evaluation component is to produce reliable usability predictions for any BOZ-designed presentation without the need for costly empirical investigation.

4.1 Perceptual Computation

The first type of utility of graphic presentations and procedures was that they sometimes reduce the amount of mental computation required to complete a task by: (a) substituting less demanding perceptual operators in place of more demanding logical operators; and (b) eliminating some perceptual operators that are not necessary to complete a task. The next two sections describe three measures of computational complexity and show how the simulation component can be used to run alternative logical and perceptual procedures produced by BOZ to generate quantitative predictions about computational differences between them.

4.1.1 Substituting Perceptual Operators

Recall that perceptual procedures are derived by substituting perceptual operators in place of logical operators in a task description. One measure of computational complexity useful for comparing informationally equivalent presentations and procedures is the *execution time* for each operator. Differences in execution time occur when a perceptual operator is performed more or less efficiently than its informationally equivalent logical operator. Differences in operator execution times can reduce the total time to complete a task, the total savings being a function of the number of times each operator is fired during performance of the task. The following definition formalizes the savings due to operator substitution.

> ***Definition 4 (savings due to operator substitution):*** *Let LOP and POP be logical and perceptual operators in logical and perceptual procedures* $P_{logical}$ *and* $P_{perceptual}$. *If* t_{LOP} *is the time required to execute LOP and* t_{POP} *the time required to execute POP, then the savings due to the operator substitution, substitution(POP, LOP), is* $n * |t_{LOP} - t_{POP}|$, *where n is the number of times LOP is executed during the procedure.*

Performance time of the perceptual operators has been shown to depend on many factors including: the skill of the user [Kieras and Polson, 1985; Hegarty and Just, 1988], practice [Schneider, 1985], particular presentation used [Lusk and Kersnick, 1979], age [Clancey and Hoyer, 1987], culture [Hudson, 1968], or even social situation [Asch, 1956]! Consequently, BOZ presently contains no way of reliably predicting when a POP for LOP substitution will turn out to yield an efficiency advantage for any particular user. For the airline reservation task, we can hypothesize that the task of perceptually estimating the distance between two flight boxes (`determineHorzDistance` and `stackHeights` operators) will be performed more efficiently than the tasks of adding and subtracting the numerically expressed departure, arrival, and cost information (`subtractTimes` and `addCosts` operators). Whether or not the operator substitutions designed by BOZ

actually yield human task performance efficiency remains an empirical question that is addressed in Chapter 6.

4.1.2 Operator Elimination: Step Skipping and Emergence

A second measure of computational efficiency is the *number of times each operator is fired.* Even though BOZ's technique for deriving perceptual procedures from logical procedures proceeds by substituting perceptual for logical operators on a one-to-one basis there are two ways in which operators can be pruned from a perceptual procedure causing the total number of perceptual operator firings to be less than the number of logical operator firings. The following reviews two basic ways in which perceptual operators can be eliminated from a perceptual procedure.

Step skipping occurs when during execution of a perceptual procedure it becomes unnecessary to perform certain perceptual operators whose corresponding logical operators are necessary for the completion of the logical procedure. For example, using the airline schedules to determine the layover between two flights it is unnecessary to determine the actual departure and arrival times of those flights. That is, it is possible to simply judge the distance between two flight boxes without worrying about where along the horizontal scale the boxes are positioned. Similarly, the heights of two boxes can be added together and the total cost determined without first determining the heights of the two individual flight boxes. The following definition formalizes the notion of savings due to step skipping.

> ***Definition 5 (savings due to step skipping):*** *Let LOP_{comp} be a computation operator: (FN arg_1 ... arg_n). Let {$LOP_{search1}$, ... $LOP_{searchN}$} be search operators that determine the arguments to LOP_{comp}. If substitution(POP_{comp}, LOP_{comp}) and substitution($POP_{searchi}$, $LOP_{searchj}$) hold for all i and j $\in$ {1 ... N} then {$POP_{search1}$, ... $POP_{searchN}$} can be eliminated.*

Emergence occurs when facts expressed in two primitive graphical languages also encode other facts expressed in a third primitive graphical language. Savings due to emergence occurs when the function computed by two computation operators is subsumed by a single lookup operator. The following definition formalizes this notion.

> ***Definition 6 (savings due to emergence):*** *Let* POP_{comp1} *and* POP_{comp2} *be perceptual operators such that* $POP_{comp1} \in PGL_1$ *and* $POP_{comp2} \in PGL_2$. POP_{comp1} *and* POP_{comp2} *can be eliminated if and only if* $\exists\, POP_{search} \in PGL_3$ *such that* $substitution(POP_{search}, POP_{comp1} \circ POP_{comp2})$ *or* $substitution(POP_{search}, POP_{comp2} \circ POP_{comp1})$ *is true.*

Emergent operators implicated in Definition 6 are called *relevant* since they result in some task performance savings. *Irrelevant* emergent operators do not compute results that contribute to completion of a task. An emergent operator is *correct* when it produces facts consistent with the set of other facts encoded in or computed from the graphic presentation, and *incorrect* otherwise [Mackinlay and Genesereth, 1985]. For example, the flight boxes in the airline graphic in Figure 24, in addition to supporting Height (cost) and Width (duration) operators, also support Area operators. Using the current interpretation of the graphic airline schedule, Area operators are both correct and irrelevant since they produce meaningful results about "cost per hour" that are not necessary for the stated task.

BOZ presently contains no mechanism for automatically identifying which perceptual operators in a perceptual procedure must be performed and which can be eliminated. This occurs because BOZ has no way of guaranteeing that the results of an operator to be eliminated will not be needed in some other part of the task procedure. Consequently, perceptual operators in a procedure must be identified and marked by hand using the following convention. Operators that can be eliminated due to step skipping are prefixed by the reserved word **SSpop**, and operators eliminated due to emergence are prefixed by

the reserved word **Epop**. The following shows how the relevant perceptual operators in the airline reservation task are marked to be eliminated due to step skipping:

```
(while (search-object-with-label FLIGHT 'pit) do
  (if (shaded? flight) then
    (read-label flight LAYOVERCITY)
    (SSpop (determine-horz-pos flight ARRIVAL))
    (while (search-object-with-label CONNECTING layovercity) do
       (if (shaded? connecting) then
         (read-label flight FINALDESTINATION)
         (if (same-labels? finaldestination 'mex) then
           (SSpop (determine-horz-pos connecting DEPARTURE))
           (determine-horz-distance departure arrival LAYOVER)
           (if (and (right-of? departure arrival)
                    (left-of? layover)) then
             (SSpop (determine-height flight COST1))
             (SSpop (determine-height connecting COST2))
             (stack-heights cost1 cost2 TOTAL)
             (if (shorter? total) then
               (repeat
                 (search-object-with-label flight SEAT1)
               (until (shaded? seat1)))
               (read-label seat1 SEATNUM1)
               (repeat
                 (search-object-with-label connecting SEAT2)
               (until (shaded? seat2)
               (read-label seat2 SEATNUM2)]
```

Figure 28: Operator Elimination in the Perceptual Airline Reservation Procedure

The simulation component counts the number of operator firings during the execution of any procedure, skipping those operators that are marked with **SSpop** and **Epop**.

4.1.3 Hypothesized Computational Advantages of the Airline Graphic

Table 9 summarizes the hypothesized computational advantages of the airline schedule graphic produced by BOZ and shown in Figure 24.

Table 9: Predicted Computational Advantages of the Airline Schedule Graphic

1. Substitutes a distance judgement (`determine-horz-distance`) in place of subtr: ..ng numerically expressed departure and arrival times (`computeLayover`).

2. The `determineDeparture` and `determineArrival` operators are unnecessary due to step skipping.

3. Substitutes a shade judgement (`shaded?`) for reading the words "ok" and "full" (`available?`).

4. Substitutes judging the combined heights of two flight boxes (`stack-heights`) for adding two numerically expressed costs (`addCosts`).

5. The `determineCost` operator is unnecessary due to step skipping.

4.2 Visual Search

The second category of advantages of graphical presentations is that graphics sometimes allow users to reduce the time they spend searching for needed information. Chapter 1 showed that search reductions offered by graphical presentations are achieved in three ways: (1) grouping related information into a single spatial locality; (2) perceptually indexing information in a graphic so that users can quickly select relevant subsets of the total set of items; and (3) parallel performance of perceptual operators that compute search constraints.

An important notion used throughout the analysis of visual search is a pointer referred to as the EYE. The *EYE* keeps track of the current focus of the user's attention within a graphic presentation. The measure of search complexity for a task and presentation then is the total number of times that EYE must change positions during performance of the task.

4.2.1. Locality

Larkin and Simon (1987) demonstrated that graphics sometimes group objects and relevant attributes into one spatial location. This is accomplished by combining the use of the various perceptual dimensions to encode several related dimensions of information in one graphical object. For example, the airline schedule uses a box to represent a flight and the

height, shading, labeling, and spatial position of the box to encode information about the times, cost, and availability of that flight.

Search reductions due to locality occur when *lookup* operators (see Table 4) execute on structured facts (see Figure 29). If lookup operators must find attribute values for a given object and some or all of those attributes occur within the same structured fact, the total number of times that EYE must change positions is equal to the number of structured facts searched and not the number of individual facts considered. The following definition formalizes the notion of search savings due to locality.

> ***Definition 7 (savings due to locality):*** *Let POP_{lookup} be a perceptual lookup operator and F be a structured fact containing facts about graphical object, g: $((PGL_1\ g\ value_1), (PGL_2\ g\ value_2)\ ..., (PGL_N\ g\ value_N))$. The search cost of POP_{lookup} is 0 if EYE = F, and 1 if EYE ≠ F.*

For example, the structured factbase representation of the graphic airline schedule is shown in Figure 29. If EYE is currently positioned on the `flight117` fact, the total number of items searched for the series of lookup operator given in Figure 30 is 1 since the EYE must only be repositioned one time between the two structured facts. Search within each structured fact is performed at no cost.

```
(((labels flight117 pittsburgh)
  (labels flight117 hou)
  (horzpos flight117 10)
  (horzpos flight117 12.83)
  (height flight117 1.79)
  (shading flight117 whiteshade))
((labels flight239 hou)
  (labels flight239 mex)
  (horzpos flight239 15.00)
  (horzpos flight239 17.25)
  (height flight239 2.39)
  (shading flight239 whiteshade)))
```

Figure 29: A Set of Structured Facts

```
(PROGN (determineDeparture 'flight117 D)
       (determineArrival 'flight117 A)
       (available? 'flight117)
       (determineDeparture 'flight239 D)
       (determineArrival 'flight239 A)
       (available? 'flight239))
```

Figure 30: A Series of Lookup Operators

The simulation component records search reductions due to locality by counting only the number of structured facts visited in the factbase when structured facts are present.

4.2.2 Indexing

Other investigators have shown that some primitive graphical languages can be processed pre-attentively [Ullman, 1984]. These graphical languages are referred to here as *indexing primitive graphical languages.* When a graphic uses an indexing primitive graphical language to encode information, the following search reduction is obtained when search operators are fired.

> ***Definition 8 (savings due to indexing):*** *Let $POP_{search}(OBJ, v)$ be a search operator that searches for a structured fact containing a single fact that satisfies $POP_{search}(OBJ, v)$. If n is the total number of structured facts in a factbase, then the search savings due to indexing is from n to m, where m is that subset of n such that each structured fact in m contains a single fact satisfying $POP_{search}(OBJ, v)$.*

For example, consider a search operator that looks for a flight with a 12:00 departure. The airline schedule graphic shown in Figure 24 uses Horizontal Position to encode departure information. Using this graphic, users can simply scan a vertical column aligned with the 12:00 label on the time scale and thus restrict their search for a flight to those flights that leave in the immediate vicinity of 12:00. Similarly, when searching for a connecting flight, the use of shading to encode availability information allows users to immediately exclude from their search all shaded flight boxes since these flights are not available.

The simulation component tracks search reductions due to indexing by partitioning the factbase (i.e., computing the sets n and m in Definition 8) prior to firing any search operator whose predicate is encoded by an indexing primitive graphical language.

4.2.3 Parallelized Operators

Reductions in the number of items considered during search can be obtained by defining parallel operators in the following way.

Definition 9 (savings due to parallel operators): *Let* $POPS_{seq} = \{pop_1, pop_2, ..., pop_z\}$ *be the complete set of operators appearing in a procedure. A parallel operator is a set:*

$POP_{par} = \{pop_a, pop_i ... pop_j\}$

such that:

(i) $pop_a, pop_i ... pop_j \in POPS_{seq}$;

(ii) all $pop \in \{pop_a, pop_i ... pop_j\}$ *are invoked given a single graphical object, g;*

(iii) pop_a *is a search operator, and* $pop_i ... pop_j$ *are lookup or computation operators.*

(iv) $|POP_{par}| \leq 4$

If the operators in $POPS_{par}$ *are performed simultaneously we obtain a search reduction from m, the total number of structured facts in the factbase, to n, the number of structured facts having individual facts that satisfy each of* $\{pop_i ... pop_j\}$.

Condition (iii) specifies Treisman and Gelade's finding that functional and spatial parallelism cannot be implemented simultaneously [Treisman and Gelade, 1977; 1980]. Condition (iv) restricts the size of a parallel operator to the estimated number of simultaneously perceivable graphical dimensions [Ericsson et al, 1980].

The simulation component contains the following simple notation for indicating that two or more perceptual operators are to be executed in parallel. Parallel operators are indicated by enclosing all constituent operator in parentheses and prefixing the list with the reserved word //**pop**. Three search strategies can be derived from the airline reservation procedure given in Figure 13 through application of the operator parallelization rule. Each successive search strategy increases the size of a parallelized operator that manipulates the flight boxes in the airline graphic. Each operator added to the //pop computes an additional search constraint and removes disqualified flight facts from the set of facts visited by the EYE. The three search strategies can be summarized as follows. **rowSearch** is the procedure shown in Figure 13 and contains no parallel operators. rowSearch proceeds by first finding a flight originating from Pittsburgh and then searching all remaining flights for a flight that connects with the originating flight that satisfies the remaining time, cost, and availability constraints. These additional constraints are checked sequentially. **rightOfSearch** finds a flight originating from Pittsburgh and then considers only those flights that depart after the originating flight arrives. rightOfSearch is the same as rowSearch, but omits consideration of flight boxes that are not to the right of the end of the current flight box. **rightOfSearchU** is identical to rightOfSearch except that only facts about available flights (unshaded boxes) are visited by the EYE. Figure 31 shows the rightOfSearchU procedure.

```
(PROCEDURE
   (while (search-object-with-label FLIGHT 'pit) do
    (if (shaded? flight) then
     (read-label flight LAYOVERCITY)
     (SSpop (determine-horz-pos flight ARRIVAL))
     (while (//pop (search-object-with-label CONNECTING layovercity)
                   (same-labels? finaldestination 'mex)
                   (NOT (shaded? connecting))
                   (determine-horz-distance departure arrival LAYOVER)
                   (right-of? departure arrival)
                   (left-of? layover)) do
            (SSpop (determine-height flight COST1))
            (SSpop (determine-height connecting COST2))
            (stack-heights cost1 cost2 TOTAL)
            (if (shorter? total) then
             (repeat
              (search-object-with-label flight SEAT1)
             (until (shaded? seat1)))
             (read-label seat1 SEATNUM1)
             (repeat
              (search-object-with-label connecting SEAT2)
             (until (shaded? seat2)
             (read-label seat2 SEATNUM2)]
```

Figure 31: The rightOfSearchU Parallelized Procedure

Note that the Height operators cannot be combined with the other parallelized operators. This combination violates Condition (iii) of Definition 9 in that it requires the simultaneous implementation of functional and spatial parallelism since the `costLessThanN?` operator receives height information from two different spatial locations.

The simulation component counts the items searched during execution of only the search operator in a parallelized operator, subtracting out those items disqualified by the remaining operators in the parallel operator. Parallelizing search procedures does not affect the counting of the number of times that the operators are executed.

4.2.4 Predicted Search Advantages of the Airline Graphic

Table 10 summarizes the search advantages of the airline schedule graphic when used for the airline reservation task.

Table 10: Predicted Search Advantages of the Airline Schedule Graphic

1. Eliminates eye movement when looking up time, city, cost, and availability information since this information is represented in the same spatial locality (a single flight box).

2. Allows the user to implement **rightOfSearch** when searching for connecting flights.

3. Allows the user to implement **rightOfSearchU** when searching for connecting flights.

Four simulations were run using two equivalent presentations of a set of airline flight facts: (1) the tabular presentation shown in Figure 32; and (2) the graphic appearing in Figure 24. The simulation results are shown in Figure 33.

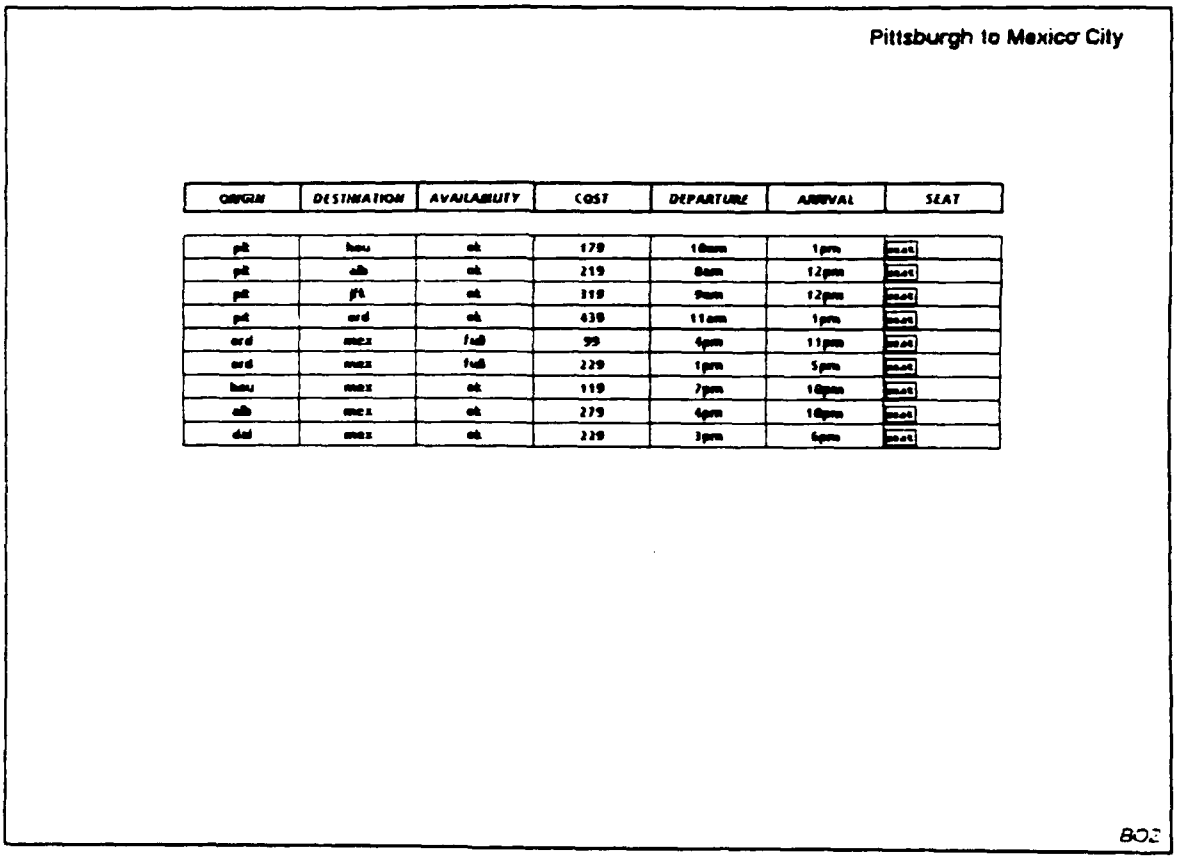
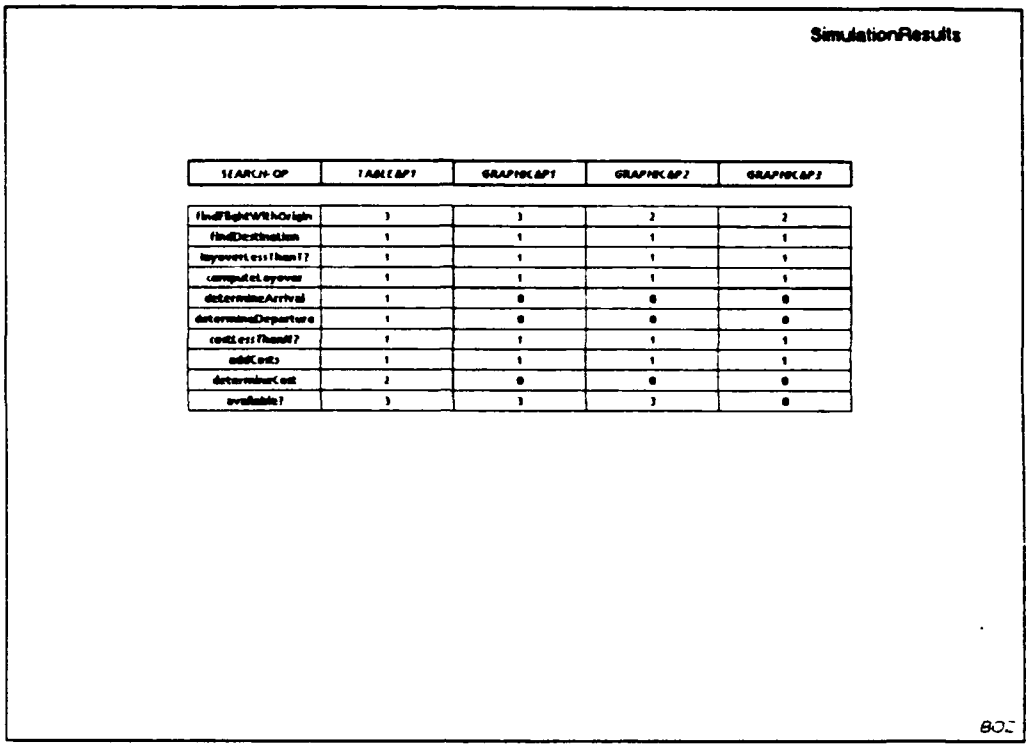


Figure 32: Tabular Airline Schedule Presentation.



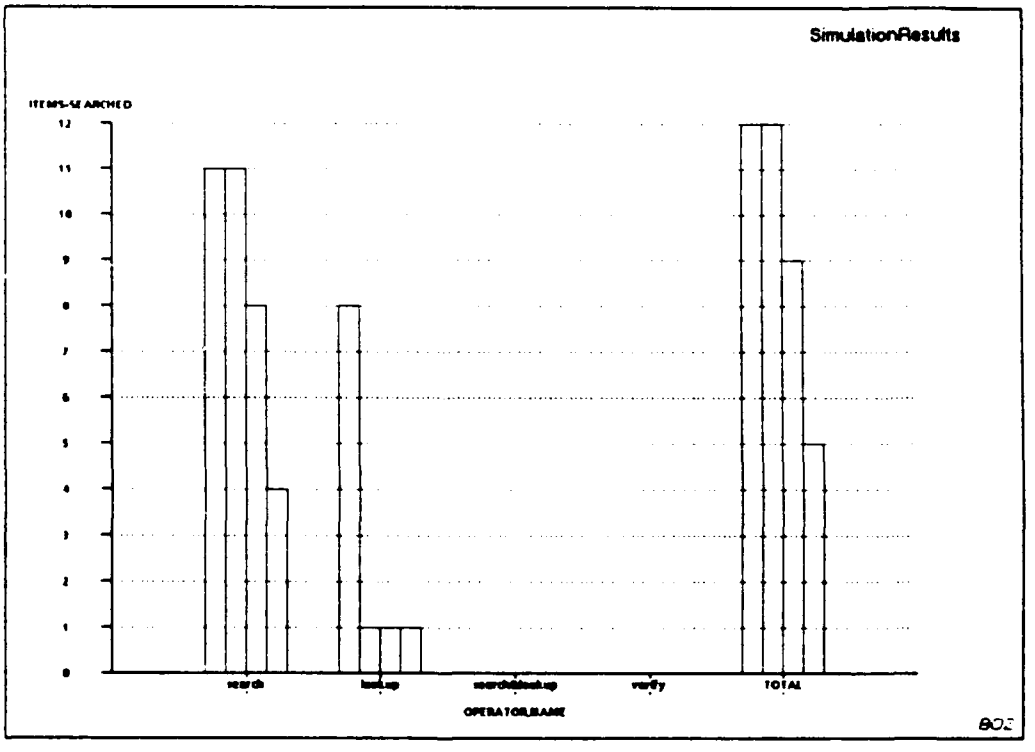


Figure 33: Simulation Results for the Alternative Procedures and Presentations.

The rowSearch procedure was run on the tabular presentation. The three alternative parallelized procedures (i.e., rowSearch, rightOfSearch, and rightOfSearchU) were run using the graphic presentation. The results show how the hypothesized advantages of the graphic presentation over the tabular presentation are captured in the simulation.

The table at the top of Figure 33 shows that when the graphic is used, the number of executions for operators that determine cost, departure, and arrival times of the flights is reduced due to step skipping. The bar chart at the bottom of Figure 33 shows the search counts for the four procedures and presentations. The first four sets of bars depict the search counts for each type of search operator individually, while the rightmost set of bars represents the total number of items searched. We can see from the first set of bars that as the perceptual procedure becomes more parallelized, the number of items considered during execution of search operators (e.g., `findFlightWithOrigin`) is reduced. This effect can be attributed to the parallelized perceptual operator shown in Figure 31 which permits a more informed search. Further search savings are achieved when lookup operators are executed (the second set of bars in Figure 33). These savings occur since the graphic airline schedule groups all information about a flight in a single graphical object. Using the tabular presentation in which information about flights is spread out across several columns, separate eye fixations are required to retrieve each value.

Note that the simulation results fail to capture two of the hypothesized advantages of the graphic presentation. First, we still have no measure of the performance time for each logical and perceptual operator, quantities upon which the predicted global savings in performance time ultimately depend. Second, there seems to be no way of determining in advance the perceptual processing skill of any individual user, and to reliably predict which, if any, of the parallel search procedures users will be able to implement. Since these two measures fall outside the scope of the predictive model they remain open for empirical investigation. In Chapter 6, observed task performance times and operator

counts are used to used to estimates these missing parameters: (a) the likelihood that a user followed a particular procedure using a particular graphic; and (b) the time required to perform each logical and perceptual operator.

CHAPTER 5

GRAPHIC DESIGN EXAMPLES

This chapter applies BOZ and the theoretical measures of graphic presentation utility to the design of graphics to support example user tasks in real-world applications. The graphics produced are not purported to be definitive solutions for each application nor do the task descriptions comprise accurate and complete characterizations of novice or expert task performance as these comprise challenging areas of investigation themselves. Rather, the examples have the following two goals: (1) to demonstrate the strengths and weaknesses of the proposed technique for transforming descriptions of logical procedures and unstructured facts to equivalent descriptions of perceptual procedures and structured graphical facts; and (2) to assess to what extent the theoretical advantages of graphic presentations are implicated by the application of this transformation to a set of example tasks. Each example contains four parts: (1) a task description; (2) a graphic presentation and perceptual procedure designed by BOZ for the task; (3) the results of a simulation that compares the BOZ-designed presentation with a conventional tabular presentation; and (4) an assessment of the effectiveness of the BOZ-designed presentation, and the ways in which the example exercises the BOZ's limitations.

5.1 An Extended Set of Airline Reservation Tasks

The following examples develop three alternative airline schedule graphics to support three other tasks that airline customers frequently perform. The three tasks can be informally summarized as follows:

(1) **Schedule a specific layover**: Many business travelers keep more than one appointment on the same trip. In this case the customer is concerned with finding a flight that passes through a particular city and lays over during a particular time period.

(2) **Find the cheapest flight(s) between two cities**: Graduate students are usually most concerned with minimizing the cost of the ticket rather than with arriving at their destination at a specific hour or day.

(3) **Find flights that minimizes the number of take-offs and landings**: Some customers try to minimize the number of take-offs and landings to minimize total travel time, the effects of jet-lag, nausea, or ear trouble.

Airline Task 1: Schedule a specific layover

Task Description. Figure 34 shows the set of logical operators for Task 1. The demanding operators in the task are the search operator that locate flights by origin, and the computation operator that computes the duration of a layover in excess of the scheduled meeting time. Operators that manipulate cost and availability information are simple lookup operators. Figure 35 shows a procedure that uses the LOPs in Figure 34 to perform the task. The procedure attempts to locate the two flights arriving in and departing from the layover city that offer the minimum amount of "down time" between the flight times and the beginning and ending time of the scheduled meeting.

```
(DOMAINSETS
 (flight NOMINAL 50)
 (origin NOMINAL (aca alb bga bue car dal gdl hou lim mex mnt oax ord
                  paz pit pva qto roi sgo))
 (destination NOMINAL (aca alb bga bue car dal gdl hou lim mex mnt oax
                       ord paz pit pva qto roi sgo))
 (departure QUANTITATIVE 100)
 (arrival QUANTITATIVE 100)
 (layovercity (departure arrival))
 (begin-meeting QUANTITATIVE 144))
 (end-meeting QUANTITATIVE 144))

(LOPS
 (NLAMBDA findFlightWithOrigin (<FLIGHT> <origin>)
  (ASK (Origin <FLIGHT> <origin>)))
 (NLAMBDA determineDestination (<flight> <DESTINATION>)
  (ASK (Destination <flight> <DESTINTATION>)))
 (NLAMBDA arrivesInLayoverCity? (<destination> <layovercity>)
  (EQUAL <destination> <layovercity>)))
 (NLAMBDA determineArrival (<flight> <ARRIVAL>)
  (ASK (Arrival <flight> <ARRIVAL>)))
 (NLAMBDA arrivesBeforeMeeting? (<arrival> <begin-meeting>)
  (LESSP <arrival> <begin-meeting>))
 (NLAMBDA computeDownTime (<arrival> <begin-meeting> <DOWNTIME>)
  (DIFFERENCE <arrival> <begin-meeting> <DOWNTIME>))
 (NLAMBDA lessDownTime? (<downtime> <downtime>)
  (LESSP <downtime> <downtime>))
 (NLAMBDA landsInDestinationCity? (<destination> <destination>)
  (EQUAL <destination> <destination>)))
 (NLAMBDA determineDeparture (<flight> <DEPARTURE>)
  (ASK (Departure <flight> <DEPARTURE>)))
 (NLAMBDA departsAfterMeeting? (<departure> <end-meeting>)
  (LESSP <departure> <end-meeting>))
 (NLAMBDA computeTotalDownTime (<downtime> <downtime> <DOWNTIME>)
  (DIFFERENCE <downtime> <downtime> <DOWNTIME>))
```

Figure 34: Logical Operators for Airline Reservation Task 1.

```
(LAMBDA (originCITY layoverCITY destinationCITY beginMEETING endMEETING)
 (repeatuntil (findFlightWithOrigin FLIGHT1 originCITY)
  (determineDestination flight1 DESTINATION1)
  (if (arrivesInLayoverCity? destination1 layoverCITY)
   then (determineArrival flight1 ARRIVAL1)
    (if (arrivesBeforeMeeting? arrival1 beginMEETING)
     then (computeDownTime beginMEETING arrival1 DOWNTIME)
      (if (lessDownTime? downtime *TIGHTEST1*)
       then (SETQ *TIGHTEST1* downtime)))))
 (repeatuntil (findFlightWithOrigin FLIGHT2 layoverCITY)
  (determineDestination flight2 DESTINATION2)
  (if (landsInDestinationCity? destination2 destinationCITY)
   then (determineDeparture flight2 DEPARTURE2)
    (if (departsAfterMeeting? departure2 endMEETING)
     then (computeDownTime departure2 endMEETING DOWNTIME)
      (if (lessDownTime? downtime *TIGHTEST2*)
       then (SETQ *TIGHTEST2* downtime)))))
 (computeTotalDownTime *TIGHTEST1* *TIGHTEST2* DOWNTIME))
```

Figure 35: Logical Procedure for Airline Reservation Task 1.

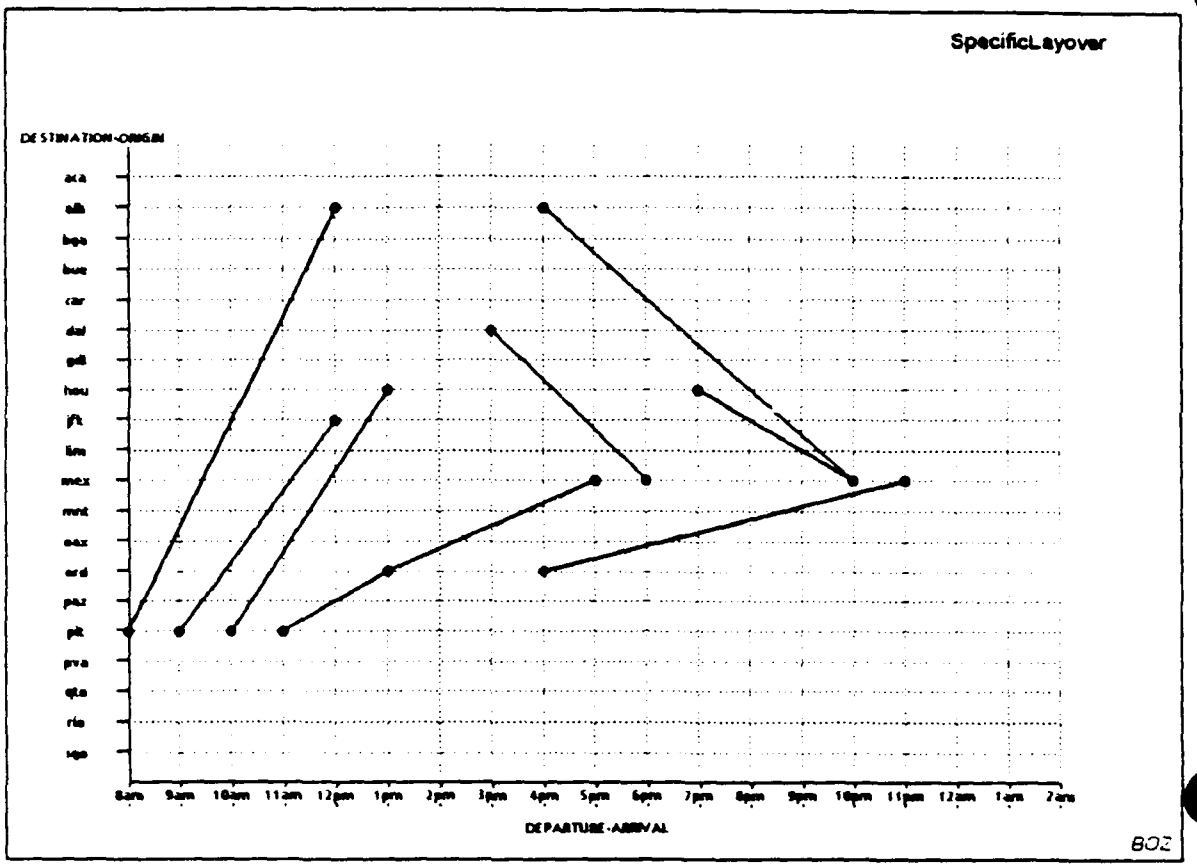


Figure 36: Graphic Presentation Designed for Airline Reservation Task 1.

Graphic Presentation and Procedure. Figure 36 shows the graphic presentation produced by BOZ. The choice of <line> as the graphical presentation object allows two identical domain sets to be composed into each axis. The horizontal axis encodes both departure and arrival time information. The horizontal positions of the two endpoints of the line encode departure and arrival times, respectively. The vertical axis encodes both origin and destination information. The vertical position of the leftmost (rightmost) endpoint encodes origin (destination).

Figure 37 shows the accompanying perceptual procedure derived by substituting the selected perceptual operators (POPs) in place of LOPs in the original logical procedure.

```
(LAMBDA (originCITY layoverCITY destinationCITY beginMEETING endMEETING)
 (repeatuntil (search-obj-at-vert-pos FLIGHT1 originCITY)
  (determine-vert-pos flight1 DESTINATION1)
  (if (vert-coincidence? destination1 layoverCITY)
   then (SSpop (determine-horz-pos flight1 ARRIVAL1))
    (if (left-of? arrival1 beginMEETING)
     then (compute-horz-distance beginMEETING arrival1 DOWNTIME)
      (if (left-of? downtime *TIGHTEST1*)
       then (SETQ *TIGHTEST1* downtime)))))
 (repeatuntil (search-obj-at-vert-pos FLIGHT2 layoverCITY)
  (determine-vert-pos flight2 DESTINATION2)
  (if (vert-coincidence? destination2 destinationCITY)
   then (SSpop (determine-horz-pos flight2 DEPARTURE2))
    (if (right-of? departure2 endMEETING)
     then (compute-horz-distance departure2 endMEETING DOWNTIME)
      (if (left-of? downtime *TIGHTEST2*)
       then (SETQ *TIGHTEST2* downtime)))))
 (horz-projection1 *TIGHTEST1* *TIGHTEST2* DOWNTIME))
```

Figure 37: Perceptual Procedure for Airline Reservation Task 1.

The procedure can be described informally as follows. To locate a flight satisfying the layover criteria the user begins by locating a flight that departs from the city of origin and arrives in the layover city. The user then determines the arrival time of the flight by determining the horizontal coincident of the right end of the line depicting the flight. The arrival time is then compared to the beginning time of the meeting. All other flights having the desired origin and destination are considered until the flight with the smallest down time is located. The earliest flight after the meeting is located analogously.

Analysis. The perceptual procedure in Figure 37 demonstrates an important limitation of the operator substitution method for generating perceptual procedures. This limitation concerns BOZ's inability to reorder the operators in a perceptual procedure to arrive at more practical perceptual procedures. A more likely perceptual procedure for the presentation in Figure 36 would be to first locate the point of intersection of the layover city and meeting time and then proceed to search for flights having the specified origins and destinations. Despite this limitation in accounting for what procedure users actually will follow, the reordered procedure produces the same simulation results as the procedure produced by BOZ. This occurs since any procedure derived through a reordering of operators has no

general efficiency advantages over another.[1] Consequently, the loss in descriptiveness does not affect the presentation produced or the measures of presentation effectiveness obtained from simulation. Chapter 7 discusses the status of BOZ-produced perceptual procedures as useful accounts of presentation-based task performance.

Table 11 informally describes the search and computation savings offered by the graphic presentation in Figure 36.

Table 11: Predicted Efficiency Advantages of the Specific Layover Graphic

SEARCH: Flights can be searched directly by layover city by locating the layover city along the horizontal axis. This search is further expedited since the cities are alphabetically ordered along the axis.
Search for a flight in a particular time interval is reduced by allowing the user to perceptually scan a horizontal interval that corresponds to that time interval.
COMPUTATION: The duration of a layover in excess of the meeting time can be determined by performing a simple horizontal distance judgement.
The operators determineDeparture and determineArrival can be eliminated due to step skipping.

The logical and perceptual procedures were used to generate quantitative predictions about the utility of the tabular airline presentation shown in Figure 33 and the graphic presentation in Figure 36. Figure 38 shows the simulation results for the alternative logical and perceptual procedures when run on a set of airline flight facts such as those shown in Figure 13. The simulation results predict that substituting the `search-obj-at-vert-pos` operator will reduce the number of items searched by one half. We can also hypothesize that the `determine-horz-distance` operator will be performed more efficiently than the `computeDownTime` operator, although this prediction cannot be made by the simulation.

[1] In general, procedures derived through operator permutations are equivalent with respect to the entire set of facts enumerable in the feature space over which the procedures are defined. However, the characteristics of a domain or task may limit the set of expressible facts to a subset of the fully enumerated fact set. In this case, efficiency differences due to operator permutations may arise.

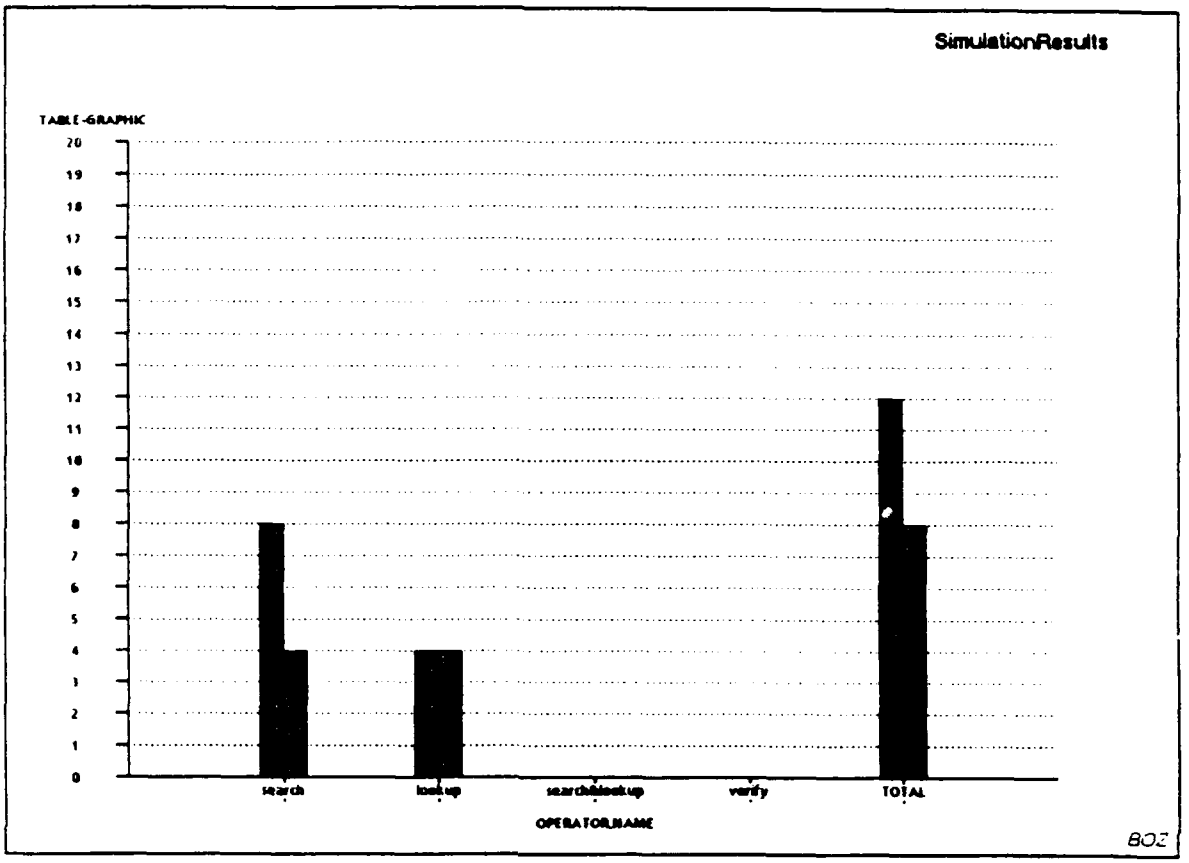


Figure 38: Simulation Results for Airline Reservation Task 1.

Airline Task 2: Find the cheapest flight(s) between two cities

Task Description. Figure 39 shows the set of logical operators for Airline Task 2. The demanding operators in this task are the search operator that searches for flights based on origin and the computation operator that computes the combined cost of two connecting flights. Figure 40 shows a procedure that uses the LOPs in Figure 39 to perform the task. The procedure searches for flights leaving from the specified city and checking the destination city. The set of flights is then searched for a flight having an origin matching the destination of the previous flight. The costs for each flight are aggregated and the lowest cost flight is chosen.

```
(DOMAINSETS
 (flight NOMINAL 50)
 (origin NOMINAL (aca alb bga bue car dal gdl hou lim mex mnt oax ord
                  paz pit pva qto roi sgo))
 (destination NOMINAL (aca alb bga bue car dal gdl hou lim mex mnt oax
                       ord paz pit pva qto roi sgo))
 (departure QUANTITATIVE 100)
 (arrival QUANTITATIVE 100)
 (cost QUANTITATIVE 500)
 (availability NOMINAL (ok full))

(LOPS
 (NLAMBDA findFlightWithOrigin (<FLIGHT> <origin>)
  (ASK (Origin <FLIGHT> <origin>)))
 (NLAMBDA determineDestination (<flight> <DESTINATION>)
  (ASK (Destination <flight> <DESTINTATION>)))
 (NLAMBDA landsInDestinationCity? (<destination> <destination>)
  (EQUAL <destination> <destination>)))
 (NLAMBDA determineCost (<flight> <COST>)
  (ASK (Cost <flight> <COST>))
 (NLAMBDA cheapestSoFar? (<cost> <cost>)
  (LESSP <cost> <cost>))
 (NLAMBDA addCosts (<cost> <cost> <COST>)
  (PLUS <cost> <cost> <COST>))
 (NLAMBDA determineArrival (<flight> <ARRIVAL>)
  (ASK (Arrival <flight> <ARRIVAL>)))
 (NLAMBDA determineDeparture (<flight> <DEPARTURE>)
  (ASK (Departure <flight> <DEPARTURE>)))
 (NLAMBDA available? (<flight> <availability>)
  (DIFFERENCE <flight> <availability>))
```

Figure 39: Logical Operators for Airline Reservation Task 2.

```
(LAMBDA (originCITY destinationCITY)
 (repeatuntil (findFlightWithOrigin FLIGHT originCITY)
  (determineDestination flight DESTINATION1)
  (if (landsInDestinationCity? destination destinationCITY)
   then (determineCost flight COST)
    (if (cheapestSoFar? cost *CHEAPESTFARE*)
     then (SETQ *FLIGHT* flight)
          (SETQ *CHEAPESTFARE* cost))))
 (repeatuntil (findFlightWithOrigin FLIGHT1 originCITY)
  (determineDestination flight1 DESTINATION1)
  (determineCost flight1 COST1)
  (repeatuntil (findFlightWithOrigin FLIGHT2 destination1)
   (determineDestination flight2 DESTINATION2)
   (if (landsInDestinationCity? destination2 destinationCITY)
    then (determineCost flight2 COST2)
     (addCosts cost1 cost2 TOTAL)
     (if (cheapestSoFar? total *CHEAPESTFARE*)
      then (SETQ *FLIGHT* (CONS flight1 flight2))
       (SETQ *CHEAPESTFARE* total))))))
```

Figure 40: Logical Procedure for Airline Reservation Task 2.

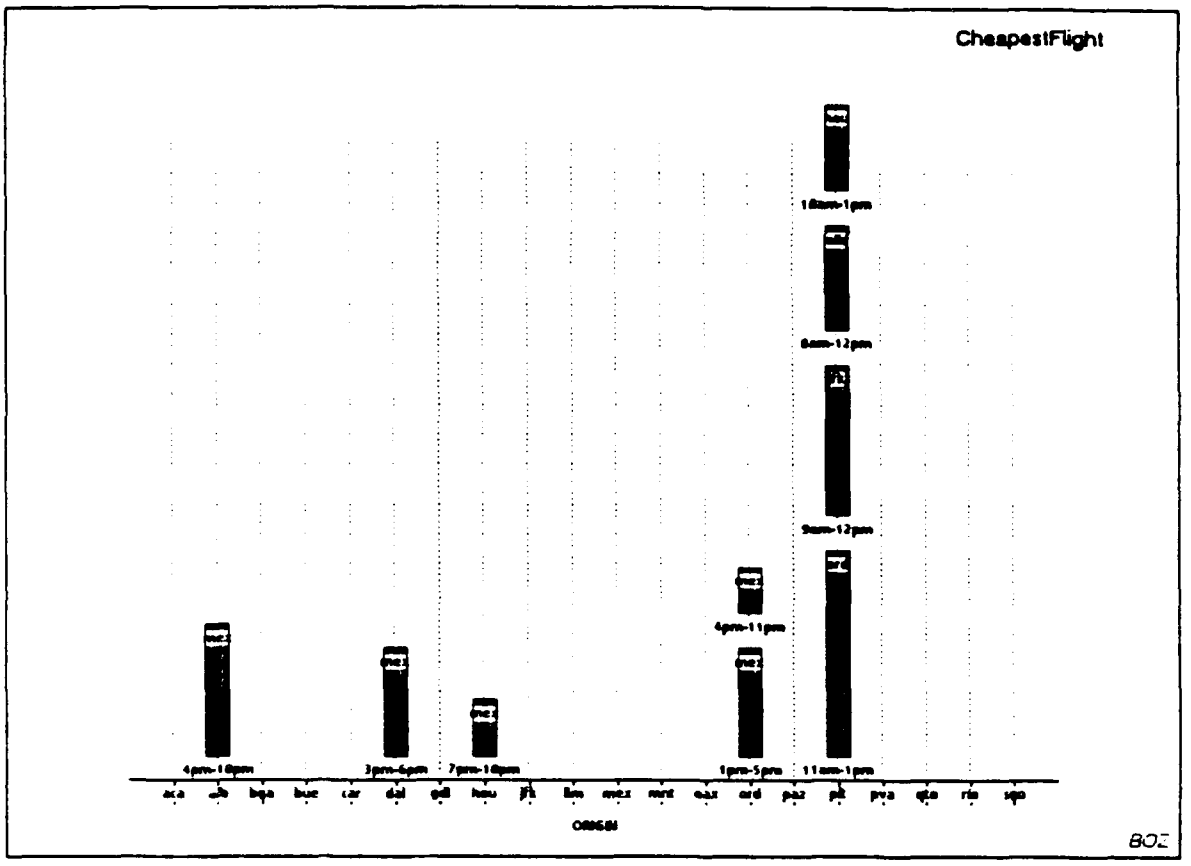


Figure 41: Graphic Presentation Designed for Airline Reservation Task 2.

Graphic Presentation and Procedure. Figure 41 shows the graphic produced by BOZ for the cheapest flight task. Figure 42 shows the accompanying perceptual procedure derived by substituting perceptual operators (POPs) in place of LOPs in the original logical procedure.

```
(LAMBDA (originCITY destinationCITY)
 (repeatuntil (search-obj-at-horz-pos FLIGHT originCITY)
  (read-label flight DESTINATION1)
  (if (same-labels? destination destinationCITY)
   then (SSpop (determine-height flight COST))
    (if (shorter? cost *CHEAPESTFARE*)
     then (SETQ *FLIGHT* flight)
          (SETQ *CHEAPESTFARE* cost))))
 (repeatuntil (search-obj-at-horz-pos FLIGHT1 originCITY)
  (read-label flight1 DESTINATION1)
  (SSpop (determine-height flight1 COST1))
  (repeatuntil (search-obj-at-horz-pos FLIGHT2 destination1)
   (read-label flight2 DESTINATION2)
   (if (same-labels? destination2 destinationCITY)
    then (SSpop (determine-height flight2 COST2))
     (stack-heights cost1 cost2 TOTAL)
     (if (shorter? total *CHEAPESTFARE*)
      then (SETQ *FLIGHT* (CONS flight1 flight2))
       (SETQ *CHEAPESTFARE* total))))))
```

Figure 42: Perceptual Procedure for Airline Reservation Task 2.

The procedure can be described informally as follows. The user must locate a flight box with the specified city of origin along the horizontal axis. The destination city is determined by reading the label on the flight box. The destination city information is then used to locate a flight that has a matching origin by searching along the same horizontal axis. For each connecting flight found the combined cost of the two flights is determined by judging the combined heights of the two flight boxes. Among these combinations the user must choose the two flight boxes having the smallest combined height.

Analysis. Table 12 informally describes the search and computation savings offered by the graphic presentation in Figure 41.

Table 12: Predicted Efficiency Advantages of the Cheapest Flight Graphic

SEARCH: Flights can be searched directly by origin by locating the cities along the horizontal and vertical axes. This search is further expedited since the cities are alphabetically ordered along the axes.
COMPUTATION: The addCosts operator is replaced by the perceptual task of judging the combined height of two boxes.
The determineCost operator is eliminated due to step skipping.

The logical and perceptual procedures were used to generate quantitative predictions about differences between the tabular presentation shown in Figure 33 and the graphic presentation in Figure 41. Figure 43 shows the simulation results for the alternative logical and perceptual procedures when run on the set of flight facts given in Figure 13. Since the search component of the task inherently requires backtracking, indexing the flights by city causes the number of items searched to be dramatically reduced. Savings due to locality (lookup operators) occur since all information pertaining to each flight is available at a single location.

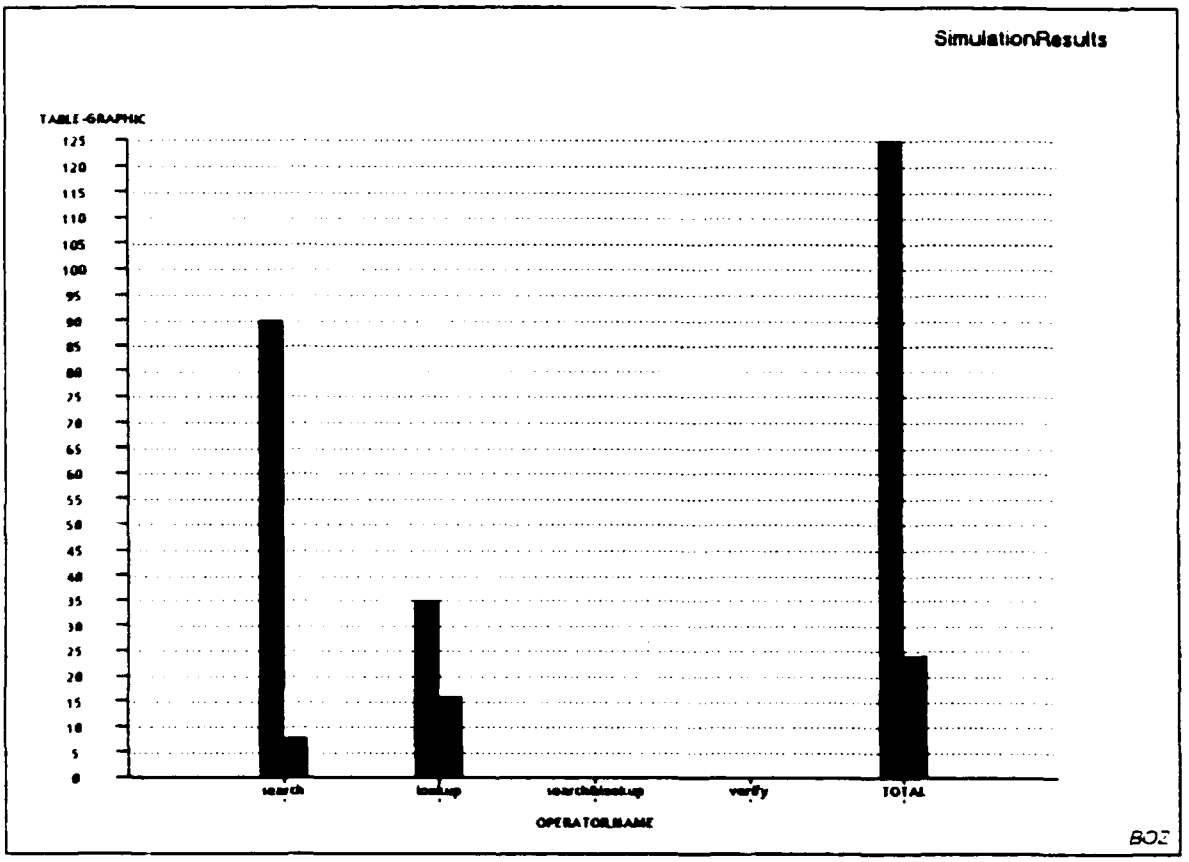


Figure 43: Simulation Results for Airline Reservation Task 2.

As an exercise we can simplify the task by requiring the user to locate a single flight between two specified cities having the lowest cost, between Pittsburgh and Houston for

example. Removing the `addCosts` operator from the LOPs description in Figure 39 accomplishes this. Figure 44 shows the graphic produced by BOZ for this task.

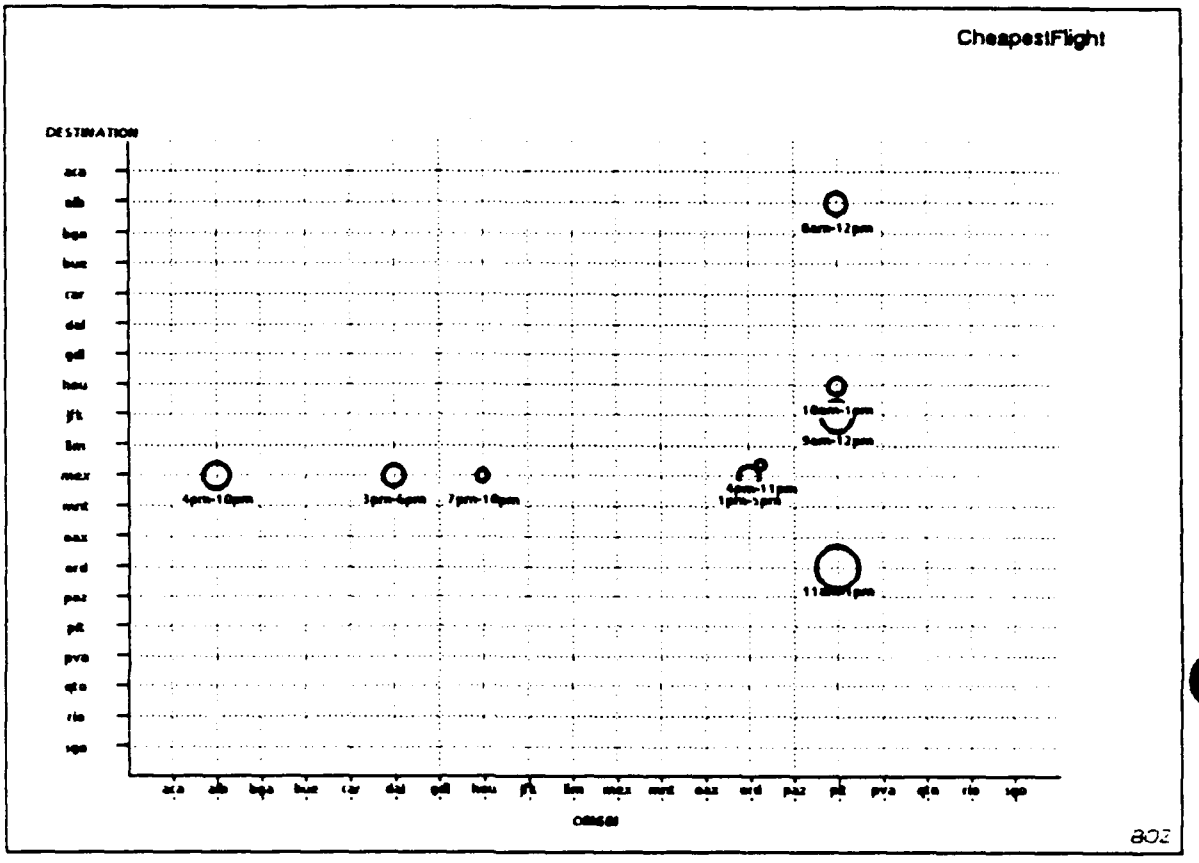


Figure 44: Alternative Graphic Presentation for Airline Reservation Task 2.

Note that by lowering the task requirements on the representation of cost information allows a less powerful primitive graphical language that does not support arithmetic operators, namely Area, to qualify. Additionally, avoiding the use of Height to encode cost information makes the vertical dimension available to encode other information. In this case, the destination of the flights is encoded along the vertical axis. Searching for a flight between two cities now reduces to finding the point of intersection for a vertical and horizontal position.

Airline Task 3: Minimize Number of Connections

Task Description. Figure 45 shows the set of logical operators for Airline Task 3. The demanding operator in the task is the search operator that locate flights by origin. Operators that manipulate time, cost, and availability information are simple lookup operators. Figure 46 shows a procedure that uses the LOPs in Figure 45 to perform the task.

```
(DOMAINSETS
  (flight NOMINAL 50)
  (origin NOMINAL (aca alb bga bue car dal gdl hou lim mex mnt oax ord
                   paz pit pva qto roi sgo))
  (destination NOMINAL (aca alb bga bue car dal gdl hou lim mex mnt oax
                        ord paz pit pva qto roi sgo))
  (departure QUANTITATIVE 100)
  (arrival QUANTITATIVE 100)
  (cost QUANTITATIVE 500)

(LOPS
  (NLAMBDA findFlightWithOrigin (<FLIGHT> <origin>)
    (ASK (Origin <FLIGHT> <origin>)))
  (NLAMBDA determineDestination (<flight> <DESTINATION>)
    (ASK (Destination <flight> <DESTINTATION>)))
  (NLAMBDA landsInDestinationCity? (<destination> <destination>)
    (EQUAL <destination> <destination>)))
  (NLAMBDA determineCost (<flight> <COST>)
    (ASK (Cost <flight> <COST>))
  (NLAMBDA determineArrival (<flight> <ARRIVAL>)
    (ASK (Arrival <flight> <ARRIVAL>)))
  (NLAMBDA determineDeparture (<flight> <DEPARTURE>)
    (ASK (Departure <flight> <DEPARTURE>)))
  (NLAMBDA connecting? (<departure> <arrival>)
    (GREATERP <departure> <arrival>))
```

Figure 45: Logical Operators for Airline Reservation Task 3.

```
(LAMBDA (originCITY destinationCITY)
 (SETQ DONE NIL)
 (repeatuntil (OR DONE (findFlightWithOrigin FLIGHT originCITY))
  (determineDestination flight DESTINATION)
  (if (landsInDestinationCity? destination destinationCITY)
   then (determineCost flight COST)
        (determineDeparture flight DEPARTURE)
        (determineArrival flight ARRIVAL)
        (SETQ DONE T)))
 (repeatuntil (OR DONE (findFlightWithOrigin FLIGHT1 originCITY))
  (determineDestination flight1 DESTINATION1)
  (determineArrival flight1 ARRIVAL1)
  (repeatuntil (OR DONE (findFlightWithOrigin FLIGHT2 destination1)
   (determineDestination flight2 DESTINATION2)
   (if (landsInDestinationCity? destination2 destinationCITY)
    then (determineDeparture flight2 DEPARTURE2)
      (if (connecting? departure2 arrival1)
       then (determineCost flight1 COST1)
        (determineCost flight2 COST2)
        (SETQ DONE T)))))
 (repeatuntil (OR DONE (findFlightWithOrigin FLIGHT1 originCITY))
  (determineDestination flight1 DESTINATION1)
  (determineArrival flight1 ARRIVAL1)
  (repeatuntil (OR DONE (findFlightWithOrigin FLIGHT2 destination1)
   (determineDeparture flight2 DEPARTURE2)
   (if (connecting? departure2 arrival1)
    then (determineDestination flight2 DESTINATION2)
      (determineArrival flight2 ARRIVAL2)
      (repeatuntil (OR DONE (findFlightWithOrigin FLIGHT3
                                                  destination2))
       (determineDestination flight3 DESTINATION3)
       (if (landsInDestinationCity? destination3 destinationCITY)
        then (determineDeparture flight3 DEPARTURE3)
          (if (connecting? departure3 arrival2)
           then (determineCost flight1 COST1)
             (determineCost flight2 COST2)
             (determineCost flight3 COST3)
             (SETQ DONE T)))))
 (LIST flight flight1 flight2 flight3)
```

Figure 46: Logical Procedure for Airline Reservation Task 3.

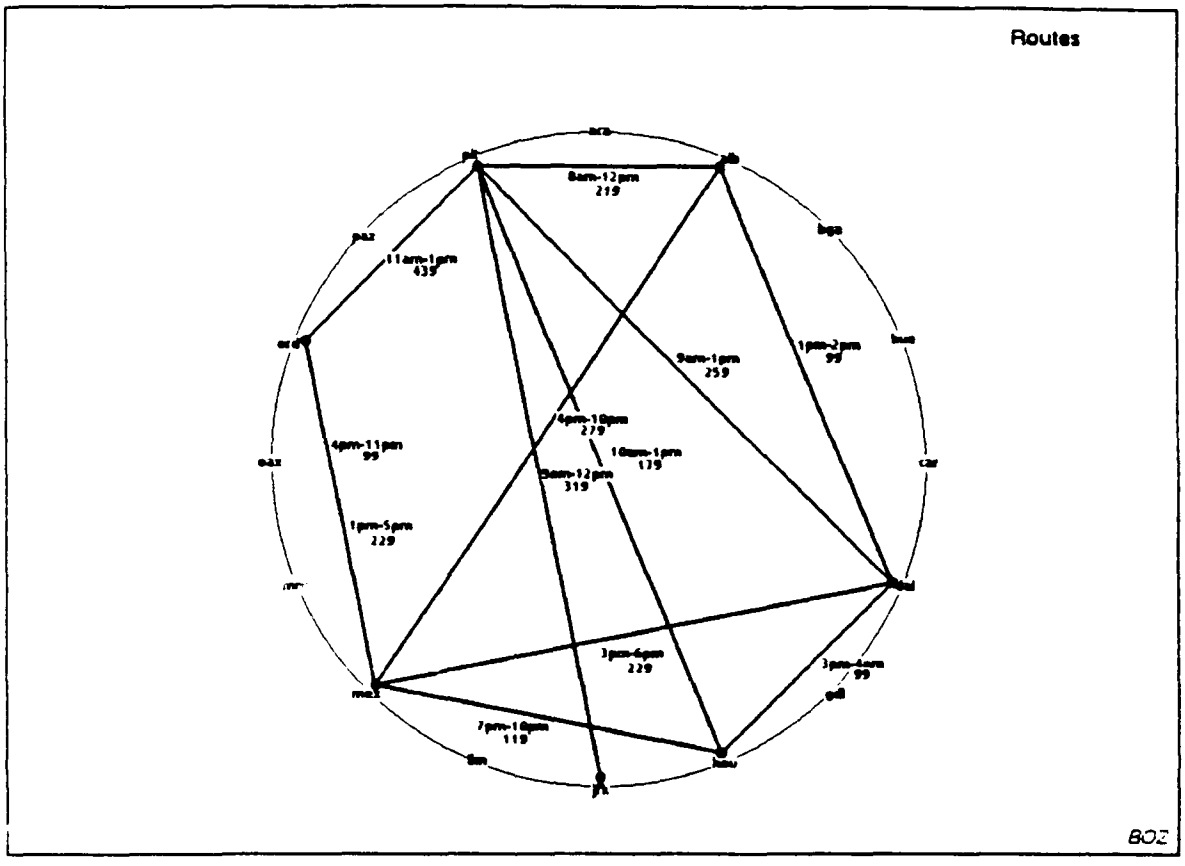


Figure 47: Graphic Presentation Designed for Airline Reservation Task 3.

Graphic Presentation and Procedure. Figure 47 shows the graphic presentation produced by BOZ. The graphic encodes each flight using a line drawn between the flight's origin and destination cities which are positioned around the circular axis. Since all other operators in the task are simple lookup operators, the information manipulated by them is encoded using labels. Figure 48 shows the accompanying perceptual procedure derived by substituting perceptual operators (POPs) in place of LOPs in the original logical procedure. Searching for a series of connecting flights follows the same procedure of locating a flight with a specific origin, determining its destination, and finding the next flight having an origin matching the destination of the previous flight.

```
(LAMBDA (originCITY destinationCITY)
 (SETQ DONE NIL)
 (repeatuntil (OR DONE (search-obj-at-horz-pos FLIGHT originCITY))
  (determine-horz-pos flight DESTINATION)
  (if (horz-coincidence? destination destinationCITY)
   then (read-label flight COST)
        (read-label flight DEPARTURE)
        (read-label flight ARRIVAL)
        (SETQ DONE T)))
 (repeatuntil (OR DONE (search-obj-at-horz-pos FLIGHT1 originCITY))
  (determine-horz-pos flight1 DESTINATION1)
  (read-label flight1 ARRIVAL1)
  (repeatuntil (OR DONE (search-obj-at-horz-pos FLIGHT2 destination1)
   (determine-horz-pos flight2 DESTINATION2)
   (if (horz-coincidence? destination2 destinationCITY)
    then (read-label flight2 DEPARTURE2)
      (if (right-of? departure2 arrival1)
       then (read-label flight1 COST1)
            (read-label flight2 COST2)
            (SETQ DONE T)))))
 (repeatuntil (OR DONE (search-obj-at-horz-pos FLIGHT1 originCITY))
  (determine-horz-pos flight1 DESTINATION1)
  (read-label flight1 ARRIVAL1)
  (repeatuntil (OR DONE (search-obj-at-horz-pos FLIGHT2 destination1)
   (read-label flight2 DEPARTURE2)
   (if (right-of? departure2 arrival1)
    then (determine-horz-pos flight2 DESTINATION2)
      (read-label flight2 ARRIVAL2)
      (repeatuntil (OR DONE (search-obj-at-horz-pos FLIGHT3
                                                    destination2))
       (determine-horz-pos flight3 DESTINATION3)
       (if (horz-coincidence? destination3 destinationCITY)
        then (read-label flight3 DEPARTURE3)
          (if (right-of? departure3 arrival2)
           then (read-label flight1 COST1)
                (read-label flight2 COST2)
                (read-label flight3 COST3)
                (SETQ DONE T)))))
 (LIST flight flight1 flight2 flight3)
```

Figure 48: Perceptual Procedure for Airline Reservation Task 3.

Analysis. The graphic in Figure 47 raises another limitation of BOZ. This limitation concerns perceptual grouping effects between graphical objects. For example, a more accurate account of the perceptual procedure followed by users would incorporate a more sophisticated search for angles composed of two lines (i.e., perceptual groups) whose endpoints coincide with the desired origin and destination points rather than the simple search for individual lines. Tasks that manipulate perceptual groups fall beyond what is describable in BOZ's task description language. Recall from Section 3.6 that this basic limitation occurs because BOZ contains no notation to indicate relations between relations.

Consequently, it is important to note that BOZ's choice of the circular axes is made to minimize occlusions among single dimensional lines when the values along the axis do not need to be ordered (nominal domain sets).

Table 13 informally describes the BOZ-predicted search and computation savings offered by the circular graphic presentation in Figure 47.

Table 13: Predicted Efficiency Advantages of the Minimum Connections Graphic

SEARCH: Search for a flight having a particular origin is reduced since city names are organized along the circular (horizontal) axis.

The perceptual procedure was used to generate quantitative predictions about the utility of the graphic presentation in Figure 47 with respect to the stated task. Figure 49 shows the simulation results for the alternative logical and perceptual procedures when run on a sample set of flight facts.

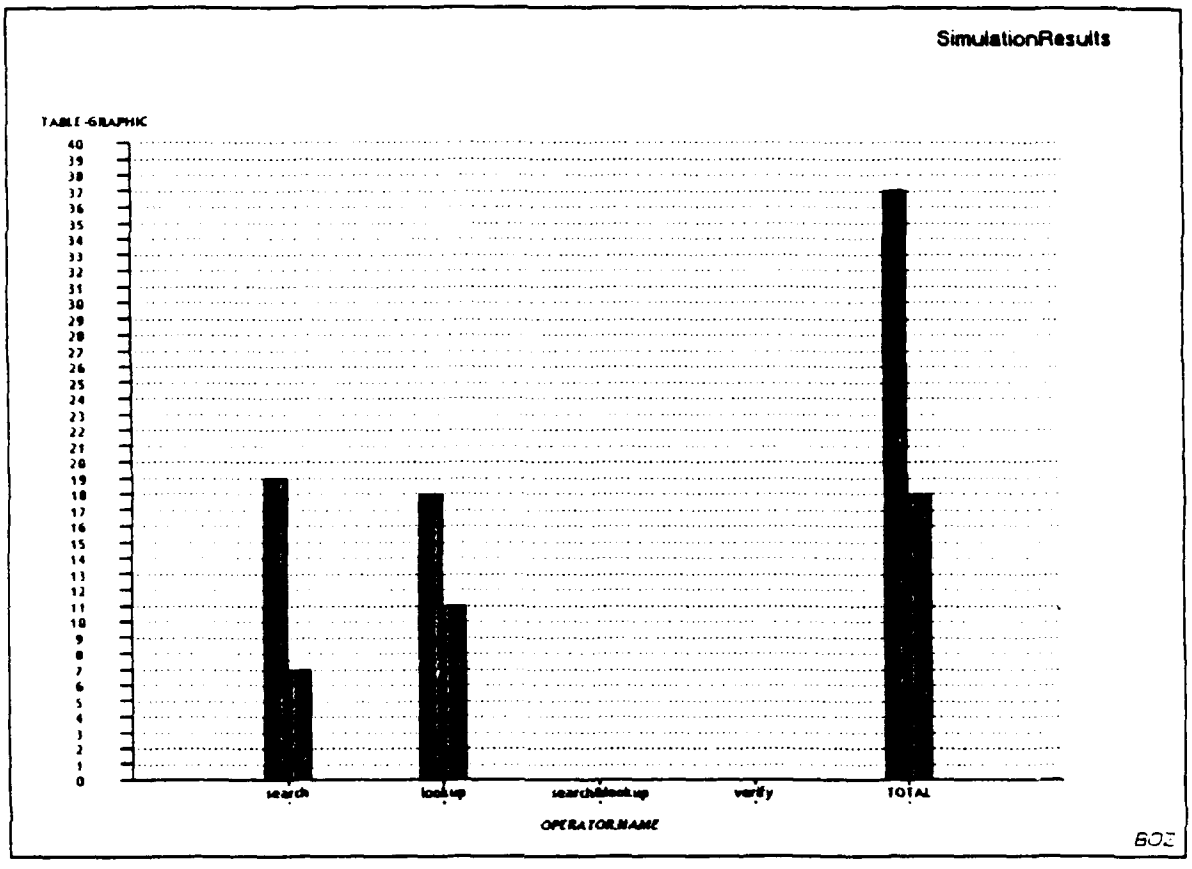


Figure 49: Simulation Results for Airline Reservation Task 3.

5.2. SPOOL File Management

In this second example the task-analytic theory is applied to a task performed by computer operators. The task involves periodically checking a list of jobs that are waiting to be assigned to printers. Before being assigned to a printer, all jobs are first sent to a single storage area called a *SPOOL file*. Print jobs are continually taken out of the SPOOL file on a first-come-first-served basis and assigned to printers. Jobs can occur in one of several paper formats: 16 x 24 inch, 10 x 12 inch, punched cards, mailing labels, etc. When a job arrives to the SPOOL file to be assigned to a printer, it must wait for a printer to become available that is currently set up to print using that particular kind of paper. Print job assignments are done automatically by the job scheduler and do not require action on the

part of the computer operator. The only action performed by the computer operator is to decide how many printers should be set up to print with each kind of paper.

A problem arises when there is a poor match between the paper formats of the jobs that come into the SPOOL file and the paper formats that are currently set up on the printers. That is, if printers are set up to print using a certain kind of paper and incoming jobs require a different format, a backlog of print jobs begins to grow in the SPOOL file while printers remain idle. Computer scientists refer to this phenomenon as poor *job throughput*. The computer operator's responsibility is to maintain a high degree of job throughput by continually monitoring a listing of the jobs currently held in the SPOOL file. When throughput decreases the operator must decide which printers should be changed to new formats to allow more jobs waiting in the SPOOL file to get through.

Task Description. Figures 50 and 51 show a set of LOPs and a procedure that performs the SPOOL task.

```
(SPOOL
   (DOMAINSETS (VALUE
    (job NOMINAL 5000)
    (jobsize QUANTITATIVE (0 128 256 512 640 768 896 1024 1152 1280))
    (jobformat NOMINAL (8x10 labels cards))
    (printer NOMINAL (spool printer001 printer002 printer003
                          printer004 printer005))
    (printerstatus NOMINAL (on off))
    (printerformat NOMINAL (8x10 labels cards))
    (jobclasssize (jobsize))
    (printerqueuesize (jobsize))

(LOPS (VALUE
   (NLAMBDA add-SPOOL-job-sizes (<jobsize> <jobsize> <JOBCLASSSIZE>)
    (PLUS <jobsize> <jobsize> <JOBCLASSSIZE>))
   (NLAMBDA overflowing? (jobclasssize> <jobclasssize>)
    (GREATERP (jobclasssize> <jobclasssize>))
   (NLAMBDA search-SPOOL-job-with-format (<JOB> <jobformat>)
    (ASK (Jobformat <JOB> <jobformat>)))
   (NLAMBDA job-waiting-in-SPOOL? (<job> <printer>)
    (ASK (Printer <job> <printer>)))
   (NLAMBDA determine-job-size (<job> <JOBSIZE>)
    (ASK (Jobsize <job> <JOBSIZE>)))
   (NLAMBDA smaller-printer-queue? (<printerqueuesize>
                                                  <printerqueuesize>)
    (LESSP <printerqueuesize> <printerqueuesize>))
   (NLAMBDA search-printer-with-format (<PRINTER> <printerformat>)
    (ASK (Printerformat <PRINTER> <printerformat>)))
   (NLAMBDA add-print-job-sizes (<jobsize> <jobsize> <PRINTERQUEUESIZE>)
    (PLUS <jobsize> <jobsize> <PRINTERQUEUESIZE>)
   (NLAMBDA search-printer-job-in-queue (<JOB> <printer>)
    (ASK (Printer <JOB> <printer>)))
   (NLAMBDA determine-format-of-printer (<printer> <PRINTERFORMAT>)
    (ASK (Format <printer> <PRINTERFORMAT>)))
   (NLAMBDA printer-on? (<printer> <printerstatus>)
    (ASK (Printerstatus <printer> <printerstatus>)))
   (NLAMBDA formats-match? (<printerformat> <jobformat>)
    (EQUAL <printerformat> <jobformat>))
   (NLAMBDA full-printer-queue? (<printerqueuesize> <printerqueuesize>)
    (GREATERP <printerqueuesize> <printerqueuesize>))
   (NLAMBDA change-printer-format (<printer> <printerformat>)
    (TELL (Printerformat <printer> <printerformat>)))
   (NLAMBDA find-off-printer (<PRINTER> <printerstatus>)
    (ASK (Printerstatus <PRINTER> <printerstatus>)))
   (NLAMBDA turn-on-printer (<printer> <printerstatus>)
    (TELL (Printerstatus <printer> <printerstatus>)))
```

Figure 50: Logical Operators for the SPOOL Task.

```
(LAMBDA
   (SETQ SPOOLCLASSES '(8x10 labels cards))
   (SETQ PRINTERS '(printer001 printer002 printer003
                                          printer 004 printer005))
   (SETQ *max-spool-capacity* 1280)
   (SETQ *max-printer-capacity* 512)
   (for spoolclass in SPOOLCLASSES
    do (SETQ TAKEACTION NIL)
       (SETQ *classtotal* 0)
       (while (search-SPOOL-job-with-format JOB spoolclass)
        do (if (job-waiting-in-SPOOL? job 'spool)
            then (determine-job-size job SIZE)
              (add-SPOOL-job-sizes size *classtotal* SPLSIZE)
              (SETQ *classtotal* SPLSIZE)))
       (if (overflowing? *classtotal* *max-spool-capacity*)
        then (SETQ TAKEACTION T))
       (if TAKEACTION
        then (SETQ machines (COPY printers))
             (while (and TAKEACTION machines)
              do (SETQ machines (CDR machines))
                (if (not (search-printer-job-in-queue JOB machine))
                 then (determine-format-of-printer machine FORM)
                   (if (not (formats-match? form spoolclass))
                    then (change-printer-format machine spoolclass))
                     (if (not (printer-on? machine 'on))
                      then (turn-on-printer machine 'on)))
                       (SETQ TAKEACTION NIL)
                      else (SETQ machines (CDR machines)))))
       (SETQ possible (COPY printers))
       (while TAKEACTION do
        (for p in possible do
         (SETQ *queuetotal* 0)
         (SETQ *smallest-queue* 10000)
         (while (search-printer-job-in-queue JOB p)
          do (determine-job-size job JSIZE)
            (add-print-job-sizes jsize *queuetotal* QSIZE)
            (SETQ *queuetotal* QSIZE)
            (if (smaller-printer-queue? qsize *smallest-queue*)
             then (SETQ *smallest-queue* *queuetotal*)
               (SETQ printer-name p)))
               (if (not (full-printer-queue?
                          *smallest-queue* *max-printer-capacity*))
                 then (determine-format-of-printer p FORMAT)
                   (if (search-printer-with-format DOUBLE format)
                    then (SETQ TAKEACTION NIL)
                      (change-printer-format double spoolclass)
                     else (SETQ possible (CDR possible)))
                  else (SETQ TAKEACTION NIL)))
       (if TAKEACTION
        then (for p in possible do
         (SETQ *queuetotal* 0)
         (SETQ *smallest-queue* 10000)
         (while (search-printer-job-in-queue JOB p)
          do (determine-job-size job JSIZE)
            (add-print-job-sizes jsize *queuetotal* QSIZE)
            (SETQ *queuetotal* QSIZE)
            (if (smaller-printer-queue? qsize *smallest-queue*)
             then (SETQ *smallest-queue* *queuetotal*)
               (SETQ printer-name p)]
```

Figure 51: Logical Procedure for the SPOOL Task.

The procedure can be summarized as follows. There are two conditions under which action must be taken: (1) when jobs of some format in the SPOOL file have grown beyond a specified limit (meaning that there are not enough printers allocated to that format); and (2) when jobs of some format have no printer allocated to it regardless of the size of that SPOOL class (starvation). There are three types of corrective actions that can be taken in either problem situation: (1) a printer can be taken from a job format that currently has several printers allocated to it but doesn't necessarily need them all; (2) a printer that is currently turned off can be turned on and formatted as required; and (3) a printer can be taken from a job format that needs it and temporarily given to a format that needs it more.

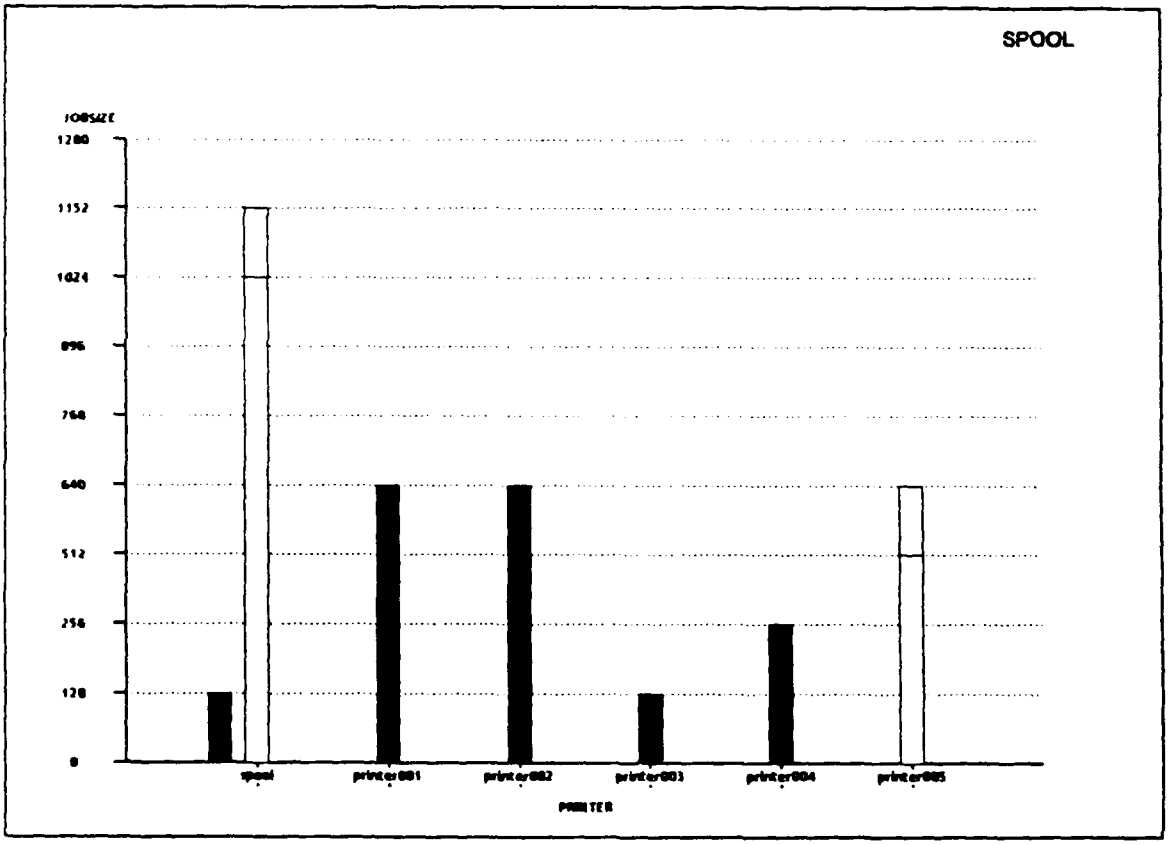


Figure 52: Graphic Presentation Designed for the SPOOL Task.

Graphic Presentation and Procedure. Figure 52 shows the graphic presentation produced by BOZ. The perceptual procedure derived by BOZ can be summarized as

follows. The user of the SPOOL graphic first scans the left of the presentation looking for stacks of bars that are higher than the 1024 limit. Such a bar indicates an overflowing set of jobs. If an overflowing stack is found, the shading of this stack is noted. Next, the user scans the right of the presentation looking for a horizontal position that contains no bar (i.e., an empty printer). If one is found, the printer corresponding to this position can be formatted to accommodate the overflowing jobs and the problem is solved. If no empty printer can be found the user must try a second solution strategy. This strategy requires the user to scan the presentation for two printers that are set up with the same format. If two printers are found the emptier of the two printers can be reformatted to the required format, leaving the second printer to process existing jobs. Finally, if no pairs of matching printers can be found, the user must simply choose the emptiest printer, reformat it, allow the overflowing jobs to process, and then change the printer back to its orignal format to avoid a second overflow.

Analysis. Table 14 informally describes the search and computation savings offered by the graphic presentation in Figure 52.

Table 14: Predicted Efficiency Advantages of the SPOOL Graphic

SEARCH: When searching for a printer that is set up to print a particular format, the user can scan for those printers whose shading matches the shading of the SPOOL class in question. When collecting all jobs of a particular format, the user can restrict their search to a single vertical stack since jobs are indexed by vertical position.
COMPUTATION: Using the stacked bar chart encoding it is unnecessary to consider and add up the sizes of the individual jobs in the queues. The user can now simply read off the height of the entire collection of jobs (the stack of rectangles) and obtain the total sum of job sizes.

The simulation tool was used to run the alternative procedures that use the tabular and graphic presentations to obtain quantitative estimates of the computational and search

complexity of each procedure and presentation. The results of the simulation are summarized in Figure 53.

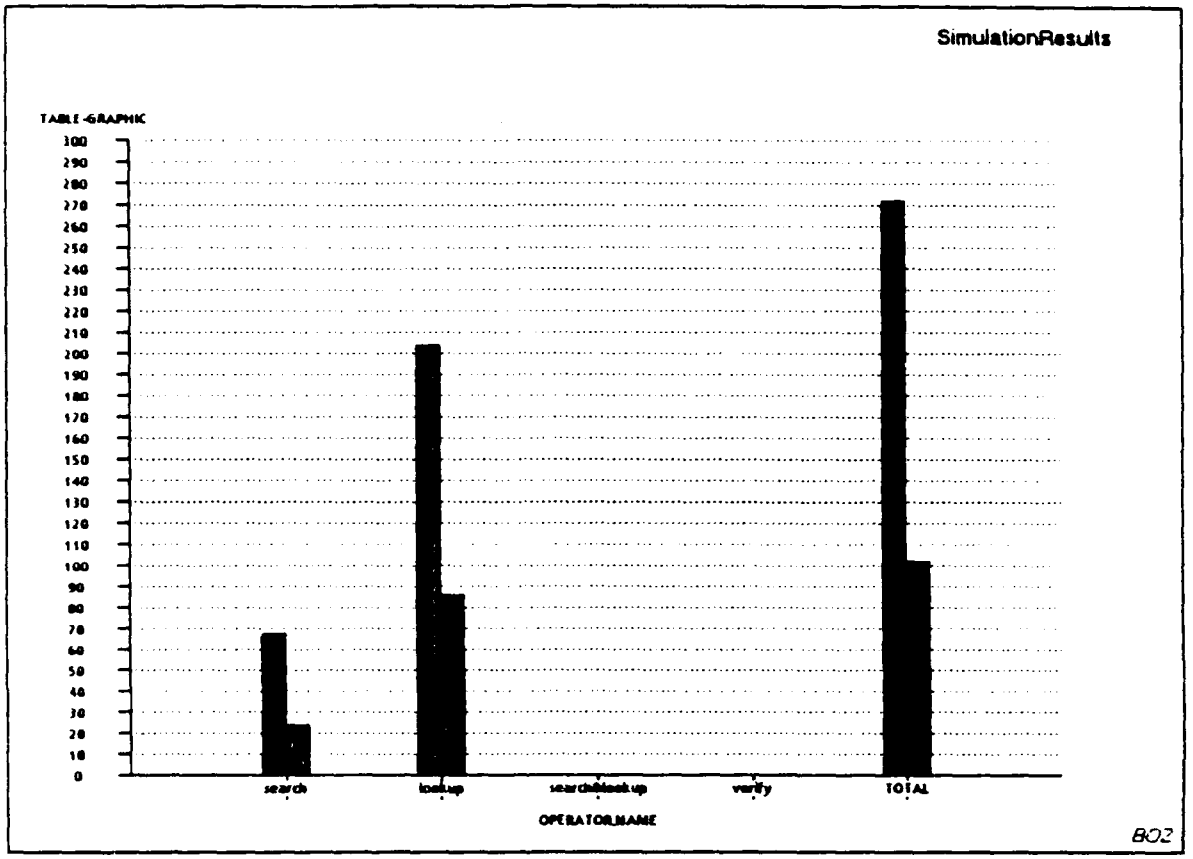


Figure 53: Simulation Results for the SPOOL Task.

Predicted search savings for the SPOOL graphic are reflected in the bar chart in Figure 53. Note that the total number of items searched using the graphic presentation is roughly one third of that for the tabular presentation. To individually measure the search savings obtained through use of the stacked bar SPOOL graphic, a simulation was run that that collects and sums SPOOL jobs separately. There was a difference of 60 items searched between the tabular and stacked bar representations of SPOOL jobs.

5.3. Class Scheduler Interface

This third application example applies BOZ to the problem of designing an interface to a computer system that allows students to schedule their own classes accessing and manipulating global information about the enrollment status of all offered classes. The design question is whether or not an interface can be designed that allows students to perform their class registration task more easily and/or successfully than they can using the traditional catalog and registration forms medium.

Task Description. The basic class scheduling task concerns choosing classes from a catalog of available classes such that a set of needed classes are located that obey day, time, and pre-requisite constraints. Many variations exist on this basic task such as locating a schedule of classes that implement morning classes only, night classes only, afternoon classes only, or 3 day per week schedules. Figure 54 shows the set of logical operators for the class scheduling tasks. The demanding operators in the task are the search operators that locate classes having a particular subject, number, and time. Figure 55 shows a procedure that uses the LOPs in Figure 54 to perform the task.

```
(ClassScheduler
(DOMAINSETS (VALUE
   (class NOMINAL 500)
   (subject NOMINAL (art biology business chemistry chinese compsci
      education french geography greek history isp japanese latin
      linguistics math philosophy physics portuguese psychology
      sociology
      spanish))
   (number ORDINAL 500)
   (starttime QUANTITATIVE 144)
   (endtime QUANTITATIVE 144)
   (classstatus NOMINAL (ok full))
   (prerequisites NOMINAL 500)
   (scheduledbystudent NOMINAL (open scheduled))))

(LOPS (VALUE
   (NLAMBDA when-does-class-begin? (<class> <STARTTIME>)
     (ASK (Starttime <class> <STARTTIME>)))
   (NLAMBDA when-does-class-let-out? (<class> <ENDTIME>)
     (ASK (Endtime <class> <ENDTIME>)))
   (NLAMBDA classes-do-not-overlap? (<starttime> <endtime>)
     (GREATERP <starttime> <endtime>))
   (NLAMBDA class-is-still-open? (<class> <CLASSSTATUS>)
     (ASK (Classstatus <class> <CLASSSTATUS>)))
   (NLAMBDA need-course-number? (<number> <number>)
     (EQUAL <number> <number>))
   (NLAMBDA sign-up-for-class (<class>)
     (TELL (Scheduledbystudent <class> T)))
   (NLAMBDA find-class-beginning-at-T (<CLASS> <starttime>)
     (ASK (Starttime <CLASS> <starttime>)))
   (NLAMBDA find-prereqs-of-class (<class> <PREREQUISITES>)
     (ASK (Prerequisites <class> <PREREQUISITES>)))
   (NLAMBDA find-class-in-subject (<CLASS> <subject>)
     (ASK (Subject <CLASS> <subject>)))
   (NLAMBDA determine-class-number (<class> <NUMBER>)
     (ASK (Number <class> <NUMBER>))))))
```

Figure 54: Logical Operators for the Class Scheduling Task.

```
(LAMBDA (courseSUBJECT courseNUMBER coursesSCHEDULED)
   (find-class-in-subject CLASS courseSUBJECT)
   (determine-class-number class NUMBER)
   (if (need-course-number? number courseNUMBER)
    then (when-does-class-begin? class BEGIN)
      (when-does-class-end? class END)
      (if (class-is-still-open? class)
       then (repeatuntil (OR (NULL coursesSCHEDULED)
              (when-does-class-begin? c BEGIN2)
              (when-does-class-let-out c END2)
              (and (classes-do-not-overlap? end begin2)
                   (classes-do-not-overlap begin end2))]
```

Figure 55: Logical Procedure for the Class Scheduling Task.

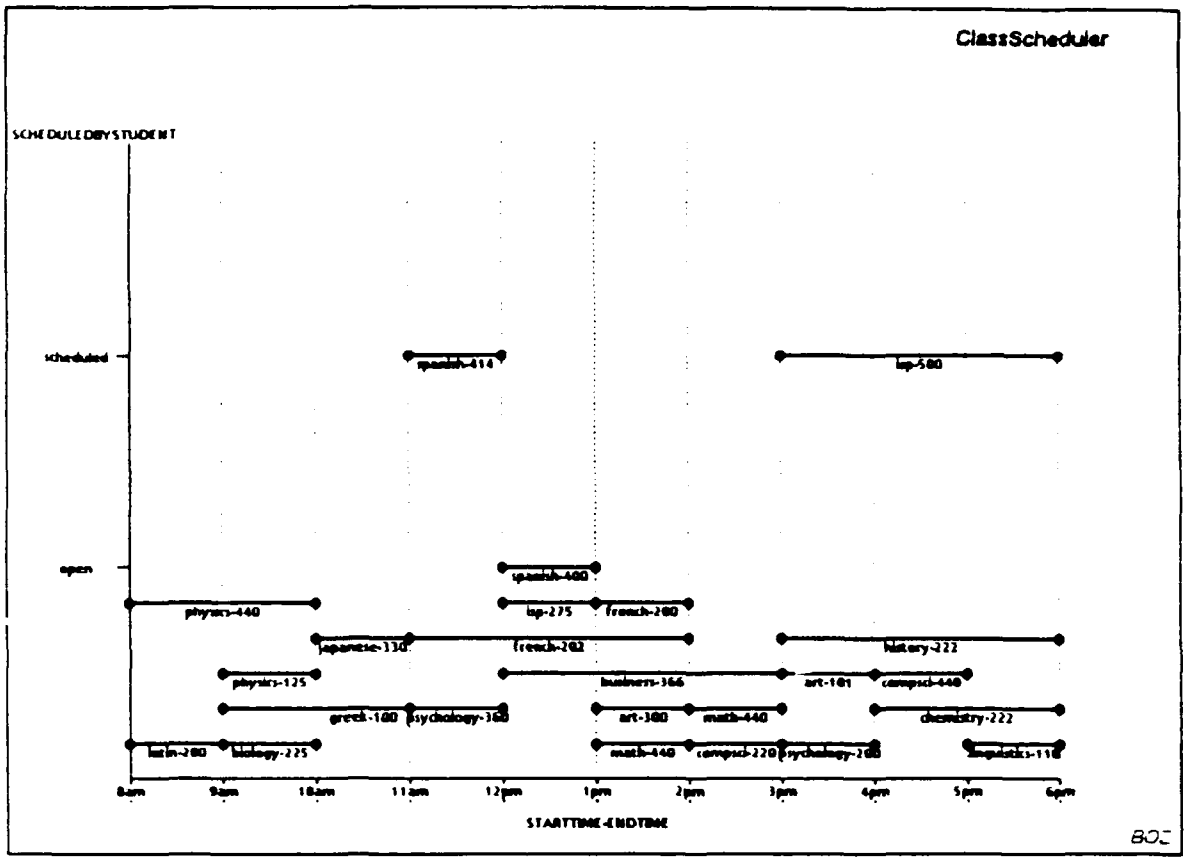


Figure 56: Graphic Presentation Designed for the Class Scheduling Task.

Graphic Presentation Display and Procedure. Figure 56 shows the graphic presentation produced by BOZ. Classes are represented by labelled lines. Class times are encoded by the left and right ends of each line. All other information pertaining to a class is encoded using labels. To sign up for a class the user can simply drag the class icon to the vertical position marked SCHEDULED BY STUDENT. Figure 57 shows the accompanying perceptual procedure derived by substituting perceptual operators (POPs) in place of LOPs in the original logical procedure.

```
(LAMBDA (courseSUBJECT courseNUMBER coursesSCHEDULED)
   (search-obj-with-label CLASS courseSUBJECT)
   (read-label class NUMBER)
   (if (same-labels? number courseNUMBER)
    then (determine-horz-pos class BEGIN)
      (determine-horz-pos class END)
      (if (visible? class)
       then (repeatuntil (OR (NULL coursesSCHEDULED)
              (determine-horz-pos c BEGIN2)
              (determine-horz-pos c END2)
              (and (left-of? end begin2)
                   (right-of? begin end2)]
```

Figure 57: Perceptual Procedure for the Class Scheduling Task.

The procedure can be described informally as follows. To locate a class having a particular subject or number, the user must search the entire presentation for an appropriately labelled line. It is important to note that the Color primitive graphical language contains operators sufficiently powerful to substitute all of the logical operators that manipulate class subject information. However, BOZ's current Xerox 1186 implementation does not have a color monitor, hence, the Color encoding is disqualified from all designs. Note that since color can be processed pre-attentively, the use of color would seem to greatly facilitate performance of the `find-class-in-subject` operator.

Analysis. This example illustrates the limits of BOZ's predictive capabilities. Table 15 informally describes the advantages of the graphic presentation in Figure 56 with respect to the task.

Table 15: Predicted Efficiency Advantages of the Class Scheduling Graphic

COMPUTATION: The user can determine whether or not two classes overlap by performing simple horizontal coincidence judgements.
SEARCH: Classes scheduled at a particular hour can be located by scanning a single column of the presentation.
Once a class has been scheduled in a particular time slot, search for other classes can rule out any available classes that fall in that time slot.

Since BOZ finds no way of supporting the `find-class-in-subject` operator, no search advantages are hypothesized. The perceptual procedure was used to generate quantitative predictions about the utility of the graphic presentation in Figure 56 with respect to the stated task. The simulation results reflect no savings in search or the number of times each operator is fired. We can still hypothesize that allowing users to perform horizontal coincidence judgements when determining time overlap between classes will be of utility, however, BOZ's simulation mechanism has no means of capturing this.

If we add the use of color to support the class subject operators we do obtain predictable search savings. Figure 58 shows the simulation results for the alternative logical and perceptual procedures when the color perceptual operators are included.

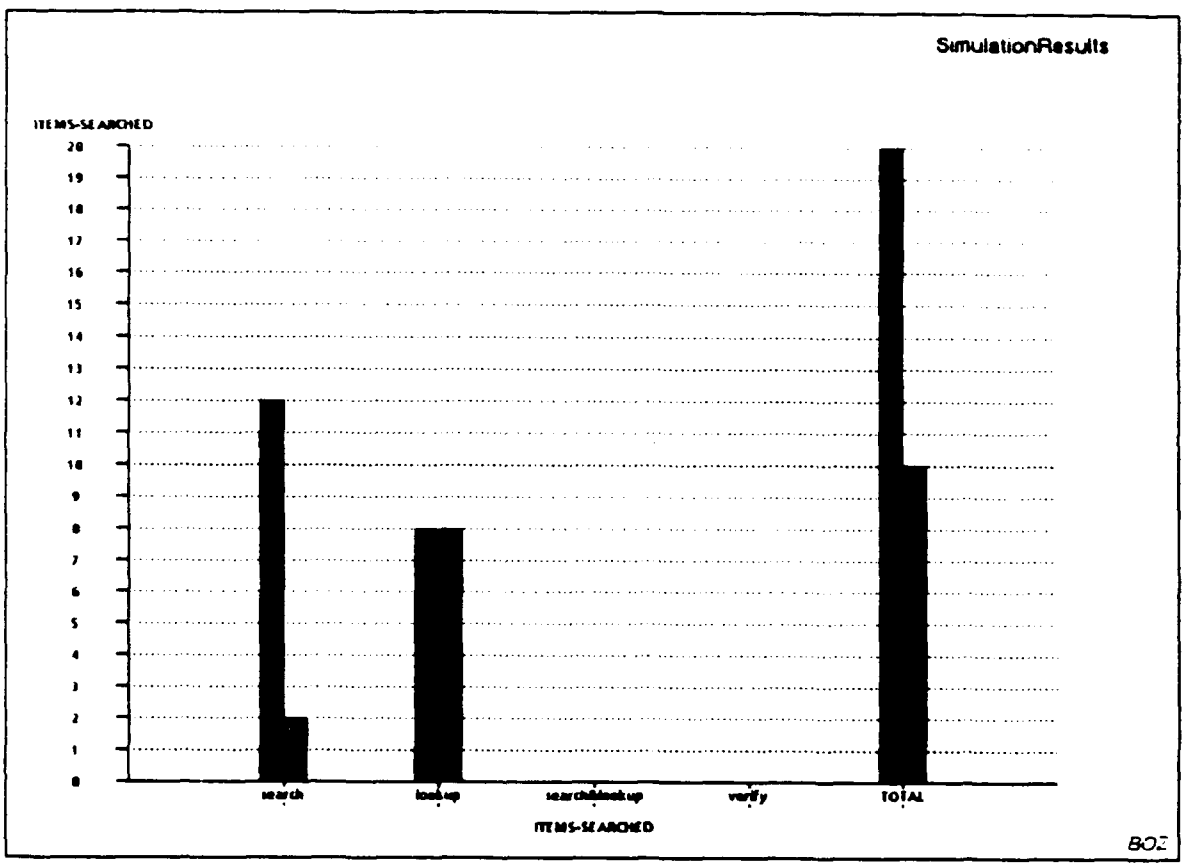


Figure 58: Simulation Results for the Class Scheduling Task.

5.4. Information Graphics

The following examples apply BOZ to the design of simple information graphics such as those presented in reports, newspapers, and magazines and that are used to convey a data set to a reader or to allow them to draw specific inferences from a set of data. Logical and perceptual procedure descriptions and quantitative predictions about the utility of the presentations are not given.

5.4.1 Consumer Report

The LOPs in Figure 59 describe a task in which the reader must select the best overall car to purchase given a set of data about the make, model, price, fuel efficiency, and reliability of a set of available cars.

```
(ConsumerReport
   (DOMAINSETS (VALUE
     (car NOMINAL 10)
     (make NOMINAL (honda nissan toyota))
     (model NOMINAL (crx dx 220SX 300SX civic grx grxII prelude accord))
     (price QUANTITATIVE 16000)
     (mbr QUANTITATIVE 24000)
     (safetydevice NOMINAL (seatbelts airbags))

(LOPS (VALUE
   (NLAMBDA determineMake (<car> <MAKE>)
     (ASK (Make <car> <MAKE>)))
   (NLAMBDA determineModel (<car> <MODEL>)
     (ASK (Model <car> <MODEL>)))
   (NLAMBDA determinePrice (<car> <PRICE>)
     (ASK (Price <car> <PRICE>)))
   (NLAMBDA cheaper? (<price> <price>)
     (LESSP <price> <price>))
   (NLAMBDA determineMPG (<car> <MPG>)
     (ASK (Mpg <car> <MPG>)))
   (NLAMBDA moreEfficient? (<mpg> <mpg>)
     (GREATERP <mpg> <mpg>))
   (NLAMBDA determineMBR (<car> <MBR>)
     (ASK (Mbr <car> <MBR>)))
   (NLAMBDA moreReliable? (<mbr> <mbr>)
     (GREATERP <mbr> <mbr>))
   (NLAMBDA determineSafetyDevice (<car> <SAFETYDEVICE>)
     (ASK (SafetyDevice <car> <SAFETYDEVICE>)))
```

Figure 59: Logical Operators for the Consumers Task.

The graphic designed by BOZ to support the task is shown in Figure 60. The perceptual procedure for finding the best car can be informally summarized as follows: "find the tallest, leftmost car that is positioned the highest along the vertical scale." This procedure can be shown to locate that car that optimizes the three evaluation criteria.

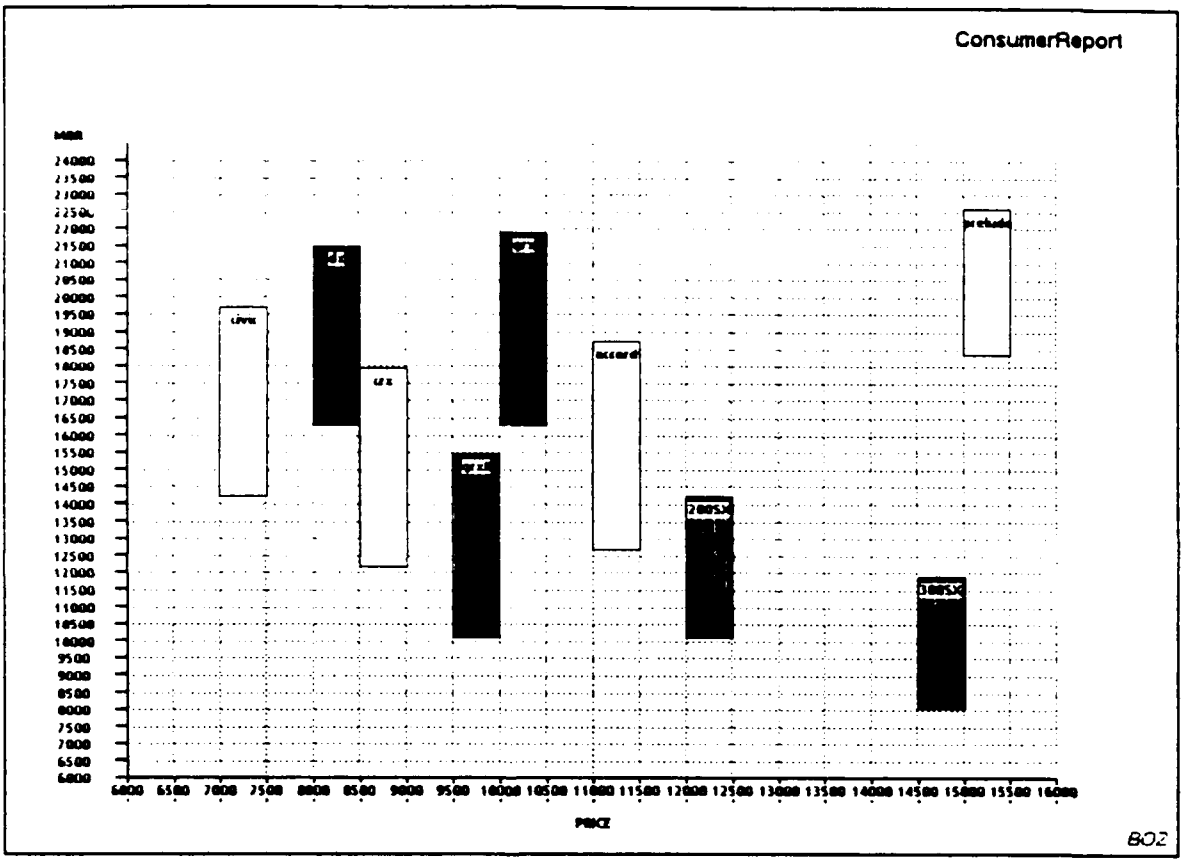


Figure 60: Graphic Presentation Designed for the Consumers Task.

The graphic in Figure 60 combines two graphical conventions that, when used together, may cause users to erroneously interpret the data. In the graphic in Figure 60, the reliability of a car (mbr: miles between repairs) is encoded by the vertical position of the bottom edge of each box. The mileage per gallon is represented by the height of the boxes. Tufte (1983) points out that readers often become confused when size dimensions are used to encode one domain set of information while the axis corresponding to that same dimension is used to encode another domain set. Users of the Consumer graphic might

conclude that the height of the box in intended to encode a *range* of values for reliability. Similarly, users might become confused when deciding where to measure the vertical position of the boxes. In other words, they may not know whether to read off the miles between repairs from the bottom or top of the boxes. This same effect is observed when the Width primitive graphical language is used together with the Horizontal Position language, and when Area is used together with Horizontal Position and Vertical Position.

Decisions about which primitive graphical languages to allow in the same presentation fall beyond the scope of BOZ. Consequently, BOZ parameterizes this design decision, allowing the user of BOZ to decide themselves by setting a global parameter. The presentation in Figure 61 is produced by BOZ when the combined use of Vertical Position and Height is disallowed.

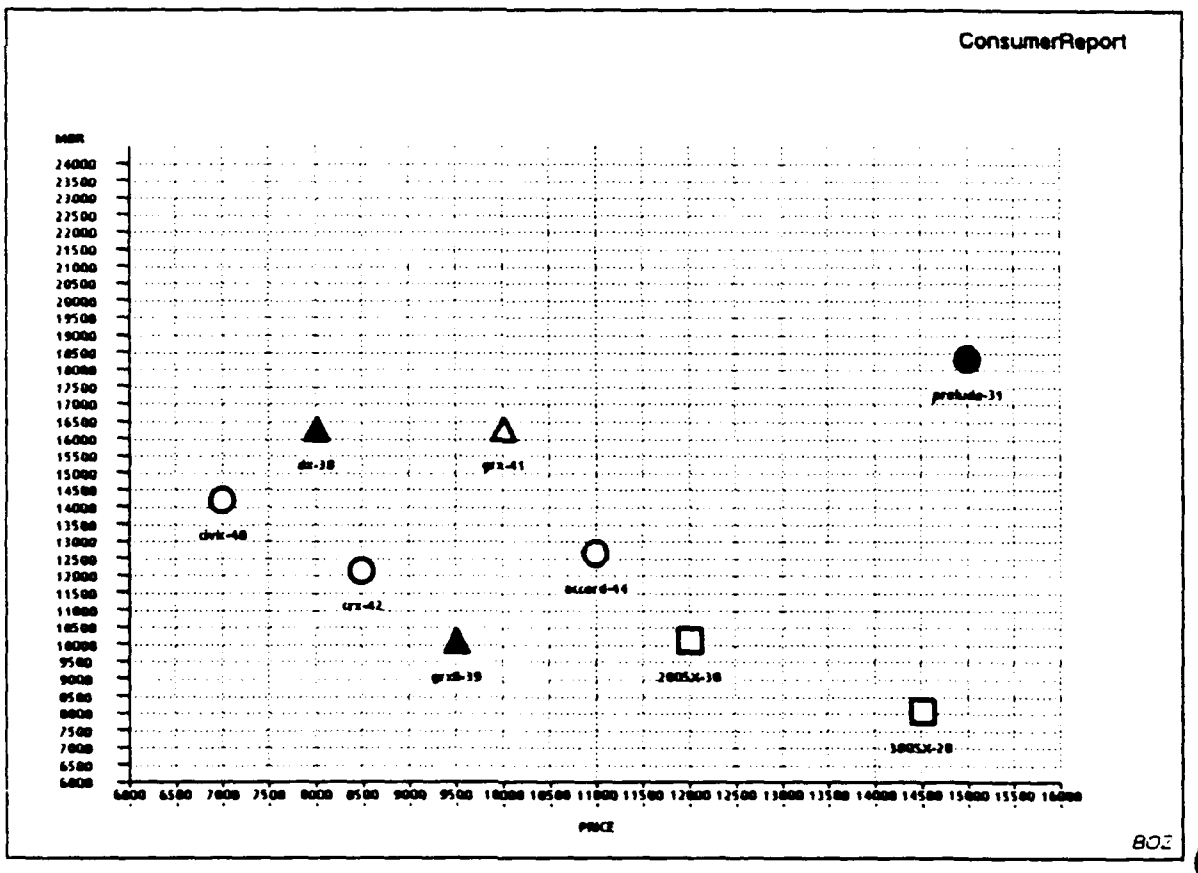


Figure 61: A Second Graphic Presentation for the Consumers Task.

5.4.2. Employees

This example illustrates how BOZ can design presentations containing more than one graphical object. Figure 62 describes a simple task of determining the names of all employees of a particular company.

```
(Employees
   (DOMAINSETS (VALUE
    (employees NOMINAL 10)
    (name NOMINAL (steve ken lael heather julia anne
                                        gale winston alison ben))
    (company NOMINAL (apple ibm xerox sun adobe quark
                                        uswest nynex nasa mcc))))

   (LOPS (VALUE
    (NLAMBDA find-any-employee-within-company (<EMPLOYEE> <company>)
     (ASK (Company <EMPLOYEE> <company>)))
    (NLAMBDA determine-employee-name (<employee> <NAME>)
     (ASK (Name <employee> <NAME>)))
```

Figure 62: Logical Operators for Employees Task 1.

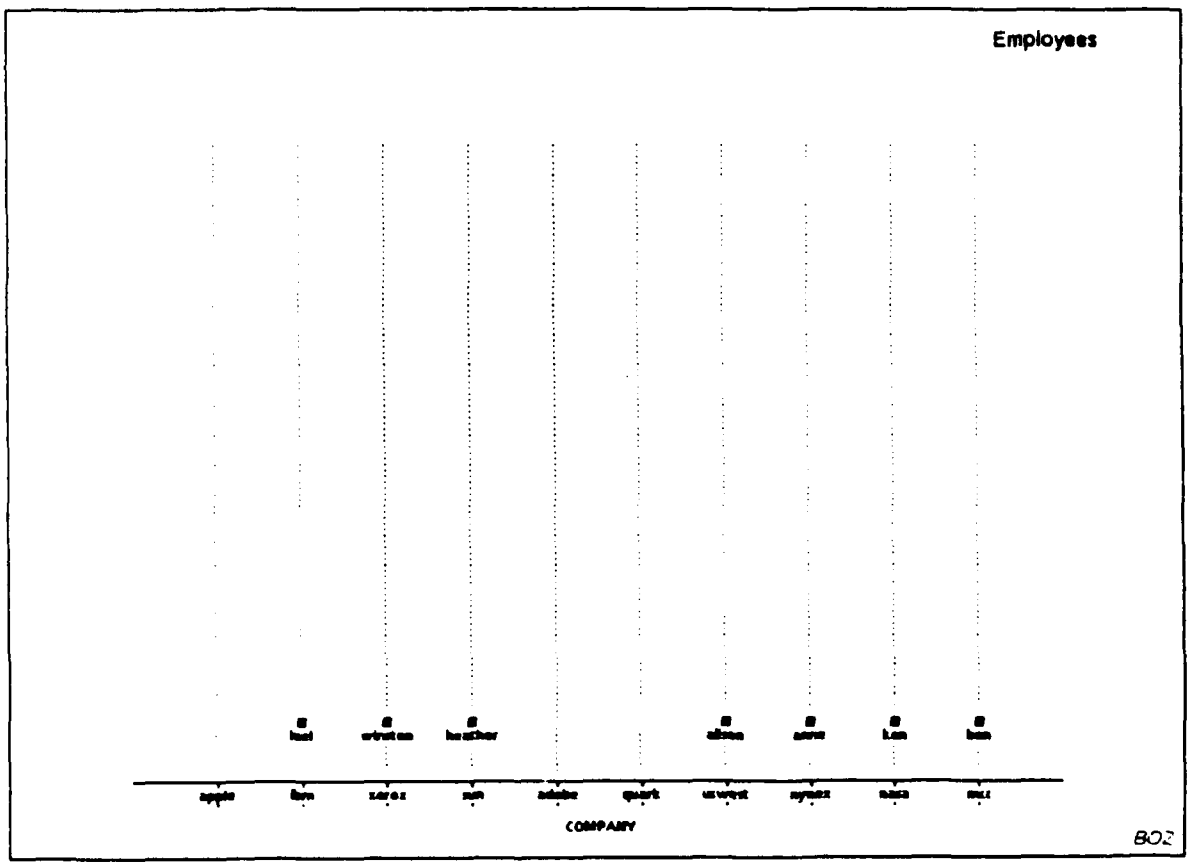


Figure 63: Graphic Presentation Designed for Employees Task 1.

Figure 63 shows the graphic produced by BOZ to support this task. Figure 64 describes a related task that requires the user to find a particular employee and report the name of the company at which the employee works.

```
(Employees
   (DOMAINSETS (VALUE
     (employees NOMINAL 10)
     (name NOMINAL (steve ken lael heather julia anne
                                        gale winston alison ben))
     (company NOMINAL (apple ibm xerox sun adobe quark
                                        uswest nynex nasa mcc))))
   (LOPS (VALUE
     (NLAMBDA find-particular-employee (<EMPLOYEE> <name>)
       (ASK (Company <EMPLOYEE> <name>)))
     (NLAMBDA determine-company-of-employee (<employee> <COMPANY>)
       (ASK (Name <employee> <COMPANY>)))
```

Figure 64: Logical Operators for Employees Task 2.

Figure 65 shows the graphic produced by BOZ to support this task.

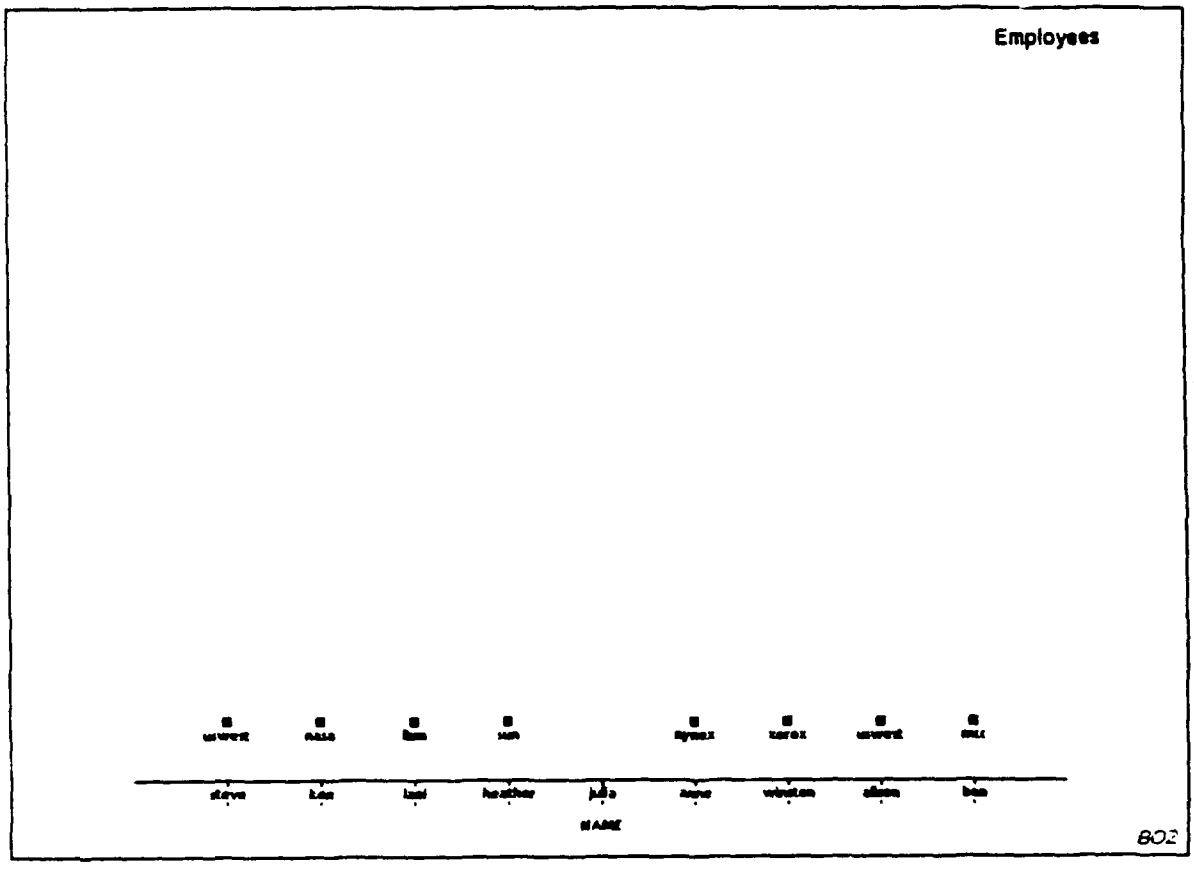


Figure 65: Graphic Presentation Designed for Employees Task 2.

The two related tasks require that the same set of information be indexed in two different ways in order to expedite search for a specific individual or company. Now suppose we combine both sets of LOPs and submit them to BOZ. This task is essentially a request for a presentation that indexes the information in both ways. Figure 66 shows the graphic produced by BOZ for this combined task.

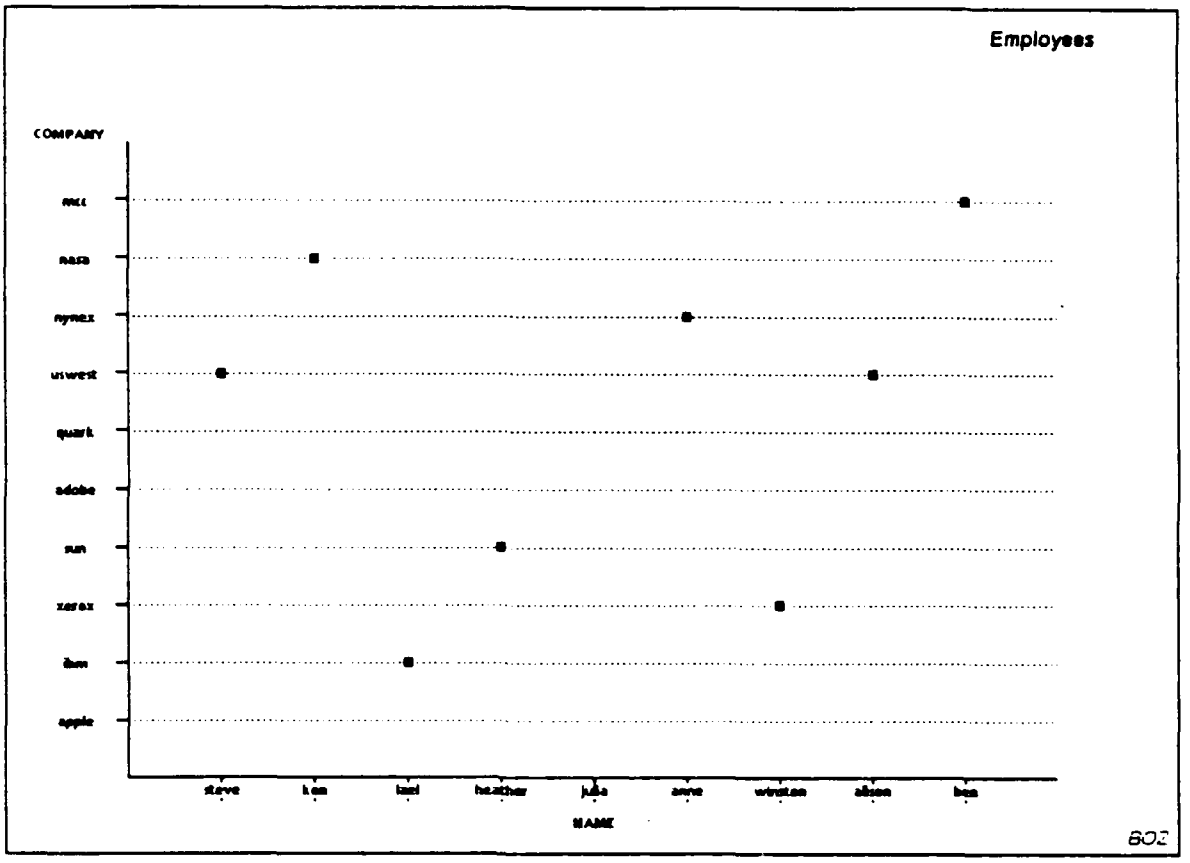


Figure 66: Graphic Presentation Designed for A Combination of Employees Tasks 1 & 2.

BOZ produces the presentation in Figure 66 by creating two separate presentations and composing them. Note that axis composition is more complicated than simply overlaying two images for two reasons. First, it must be observed that the label encodings for company and employee information appearing in the presentations in Figures 63 and 65 are subsumed by the axes of the partner presentation that it is being composed with. Second, a simple overlay of the two presentations would result in two points appearing in each entry

in the plane. Hence, a second observation is that the two points marking the positions on each axis can be merged using the graphical object composition rules given in Table 8.

5.5 Conclusion

The examples presented above are simplistic in the following way. The domain sets and logical operators appearing in each task description comprise what psychologists refer to as a *problem space* [Newell and Simon, 1972]. It is widely held that complex information-processing tasks typically draw on many problem spaces, some of them poorly understood or unidentifiable. The examples presented in this chapter are all defined using a single problem space. This was achieved in two basic ways: (1) by selecting closed-ended and reasonably well understood (e.g., the information graphics examples); or (2) by choosing restricted, well understood portions of the task (e.g., the airline scheduling and class scheduling examples). For example, the general airline scheduling problem likely considers other dimensions of information such as the customers' personal feelings about the reliability of a carrier, or uses more sophisticated search criteria that places different degrees of importance on time, cost, and place information when selecting a flight. To the extent to which the problem spaces for each task are considered simplified, the examples presented above must be considered partial or even "toy" solutions. The more general question of BOZ's usefulness for tackling many-problem space tasks must await its application to more sophisticated task analyses.

CHAPTER 6

EXPERIMENTAL STUDY OF GRAPHIC DESIGN EFFECTIVENESS

An empirical study was conducted to test the effectiveness of the task-analytic presentation design and evaluation techniques proposed in earlier chapters. Participants performed an airline reservation task using a set of four experimental presentations, each presentation including an additional opportunity to reduce search or computation. Task performance times were used as a dependent measure of the cognitive work being done while performing the task, the measure BOZ's design process aims to minimize. There were two basic goals of the experiment: (1) to measure to what extent the hypothesized advantages of the BOZ-designed airline schedule graphic were reflected in participants' performance; and (2) to attempt to understand which procedures participants followed when performing the task. Taken together, these two questions pose the more critical question of whether BOZ's design and theoretical evaluation components are presently reliable enough to proceed without the use of costly empirical evaluation.

6.1 Method

Participants. Seven employees of the Learning Research and Development Center at the University of Pittsburgh voluntarily participated in the study. Participants were not paid for their participation. The only criterion used for participant selection was that the individual had not previously been exposed to the presentations used as stimulus materials in the experiment.

Materials. There were a total of forty problems, ten problems using each of a set of four experimental presentations of airline schedule information. Examples of these

presentations are shown in Figure 67. Each presentation contains an additional opportunity to reduce computation or search. Each successive presentation contains all of the advantages of the previous ones plus an additional advantage.

Apparatus. The airline schedules were presented as 9 x 12 inch screen images on a Xerox 1186 computer. A mouse was used by the experimenter to control the presentation of the airline schedules. The mouse was also used to control the system clock that measured participants' performance times. Since participants typically traced their fingers across the computer screen when performing the task, the stopwatch was controlled by the experimenter.

Procedure. Participants performed the airline reservation task forty times, ten times using each version of the airline schedule presentation. To counterbalance learning and practice, seven rotations (one for each participant) of presentation were used (i.e., 1234, 2341, 3412, 4123, 4321, 3214, and 2143). At the start of the experiment, all of the perceptual operators were explained. Participants were shown the procedure that best exploited the advantages of each presentation (i.e., Presentation 1 = rowSearch; Presentation 2 = rightOfSearch; Presentation 3 = rightOfSearchU; Presentation 4 = rightOfSearchU)[1] but were told they could follow any procedure they wished. Participant's task was to find any flight that satisfied the problem criteria. There was one practice trial with each of the four presentations. Participants were told not to guess, and to work as quickly as possible without compromising the accuracy of their responses. Participants were told that they could rest between any two graphics and that the total time to complete the experiment was irrelevant. Time to complete the session was typically forty minutes.

[1]See Chapter 4 for descriptions of the four different procedures.

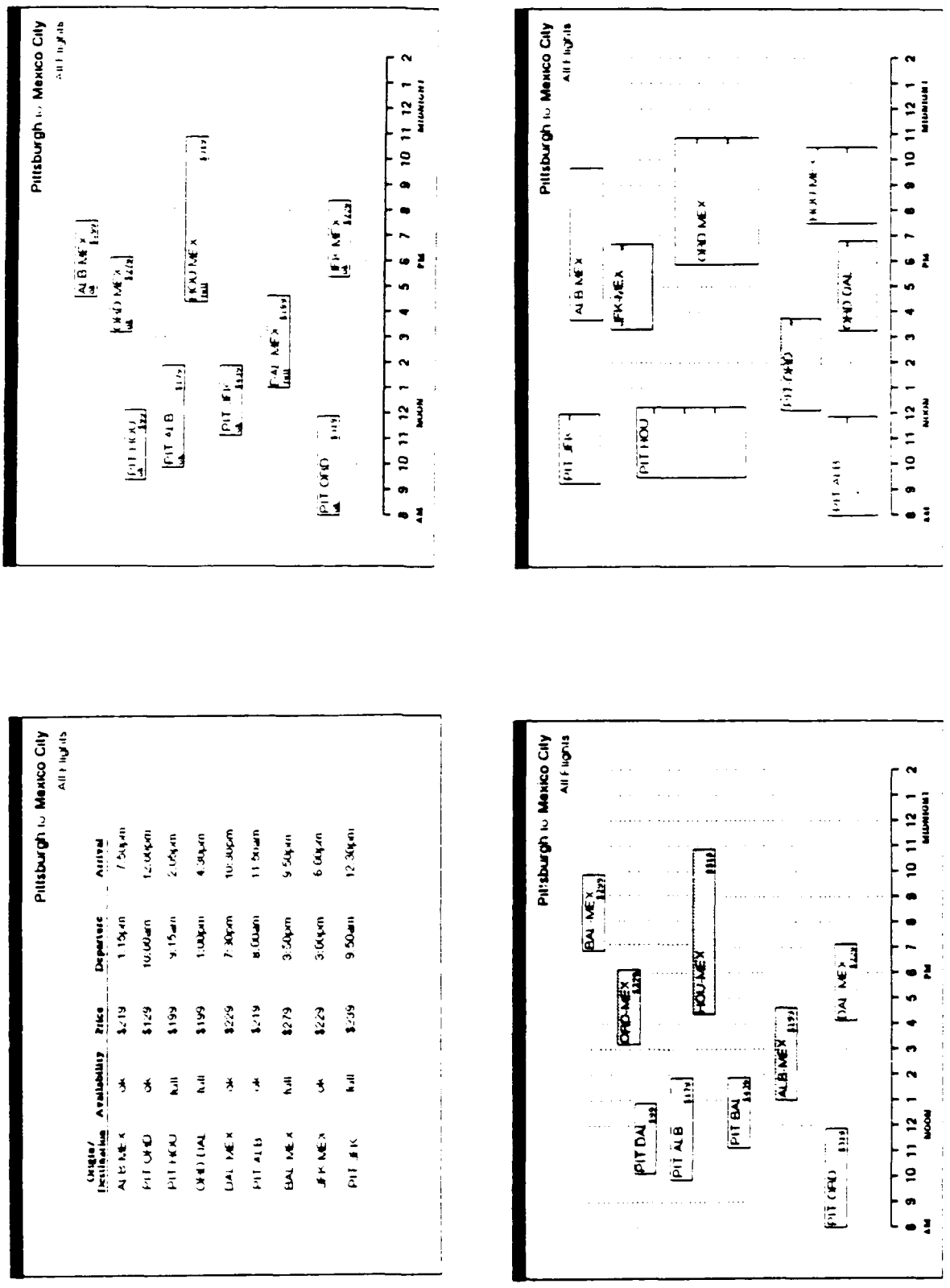


Figure 67: Four Experimental Graphics

6.2 Theoretical Predictions

Based on the analyses of the airline schedule presentations given in Chapter 4, the following predictions were made about participants' performance for the airline reservation task.

Global efficiency. Each presentation supports the advantages of the previous one as well as the ones it introduces. Therefore, the first prediction is that task performance time should be linearly ordered as in Figure 67 with the tabular presentation worst and the graphic airline schedule best.

Reductions in Computation. Recall that the simulation tool described in Chapter 4 is presently unable to produce quantitative estimates of the time required to execute each logical or perceptual operator in a task. However, if response times are expressed as a function of the number of times each operator is executed, the coefficients of a well-fitting regression model yield estimates on the performance times for each operator. It was hypothesized that the times associated with the demanding logical operators (i.e., `computeLayover` and `addCosts`) should be smaller for graphics that support substitution of visual operators. Since the `determineCost`, `determineDeparture`, and `determineArrival` operators are unnecessary for Presentations 2 through 4, these savings should be increased. Table 16 summarizes the computational advantages of the four displays.

Table 16: Predicted Computational Advantages of the Four Experimental Airline Schedules

PRESENTATION 1
Presentation 1 does not allow the user to substitute any perceptual operators.

PRESENTATION 2
Presentation 2 substitutes a horizontal distance judgement for the computeLayover logical operator. The layoverLessThanN? logical operator is substituted by the perceptual task of estimating the relative size of the horizontal interval between two flights. Performing the logical operators determineDeparture and determineArrival is unnecessary.

PRESENTATION 3
Presentation 3 additionally allows users to perform the shaded? operator in place of the determineAvailability operator.

PRESENTATION 4
Presentation 4 substitutes the perceptual task of judging the combined height of two flight boxes for the addCosts logical operator. The logical operator determineCost is unnecessary.

Reductions in Search. The simulation results predict several search reductions gained due to locality, indexing, or the use of parallel operators. Table 17 lists six hypothesized search procedures that explain the three ways in which search can be curtailed. If users are able to follow the search procedures hypothesized for each airline schedule, a regression of response times to the total number of items searched should produce good fits for procedures that users follow and poor fits for procedures not used.

Table 17: Predicted Search Advantages of the Four Experimental Airline Schedules

PRESENTATION 1
rowSearch: Spatially organizes information about flights into rows and columns of a table. When searching for information about a particular flight, the user need only locate the appropriate row and column and restrict their search to those items.

PRESENTATION 2
rightOfSearch: When searching for a connecting flight (i.e., a <city>-MEX flight), limit search to only those flight boxes appearing to the right of the originating flight. Users can also eliminate connecting flights that appear to the extreme right of the graphic since these flights obviously disobey the layover constraint.

PRESENTATION 3
rightOfSearchU: When searching for an originating or connecting flight, limit search to only unshaded flight boxes since others have no available seats.

PRESENTATION 4
rightOfCheapSearchU: When searching for a connecting flight, eliminate "tall" flights (i.e., flight boxes whose height combined with the height of the originating flight box exceed the $500 limit).

6.3 Results and Discussion

Global efficiency. Figure 68 shows the mean response time for each presentation (excluding five times differing by more than three standard deviations from the problem mean and the three to six erroneous responses for each presentation). Presentation used had a highly significant effect on response time ($F(3, 239) = 52.719$, $p < .0001$), and also on the variance of response time ($F(3, 24) = 18.649$, $p < .0001$). Fischer's PLSD for pairwise comparison indicates indicates no significant difference between version 3 and 4 and differences significant at the .05 level between other presentation pairs for both mean response times and standard errors of the mean. Presentations 3 and 4 produce both the lowest response times and the least variable performance. Presentations 2 and 1 each in turn produce significantly higher response times and greater variability. The perceptual operators and search savings supported by Presentations 2 and 3 together had the predicted effect on global efficiency. But allowing users to perform `judgeHeights` (instead of `addCosts`) produced no observable effect.

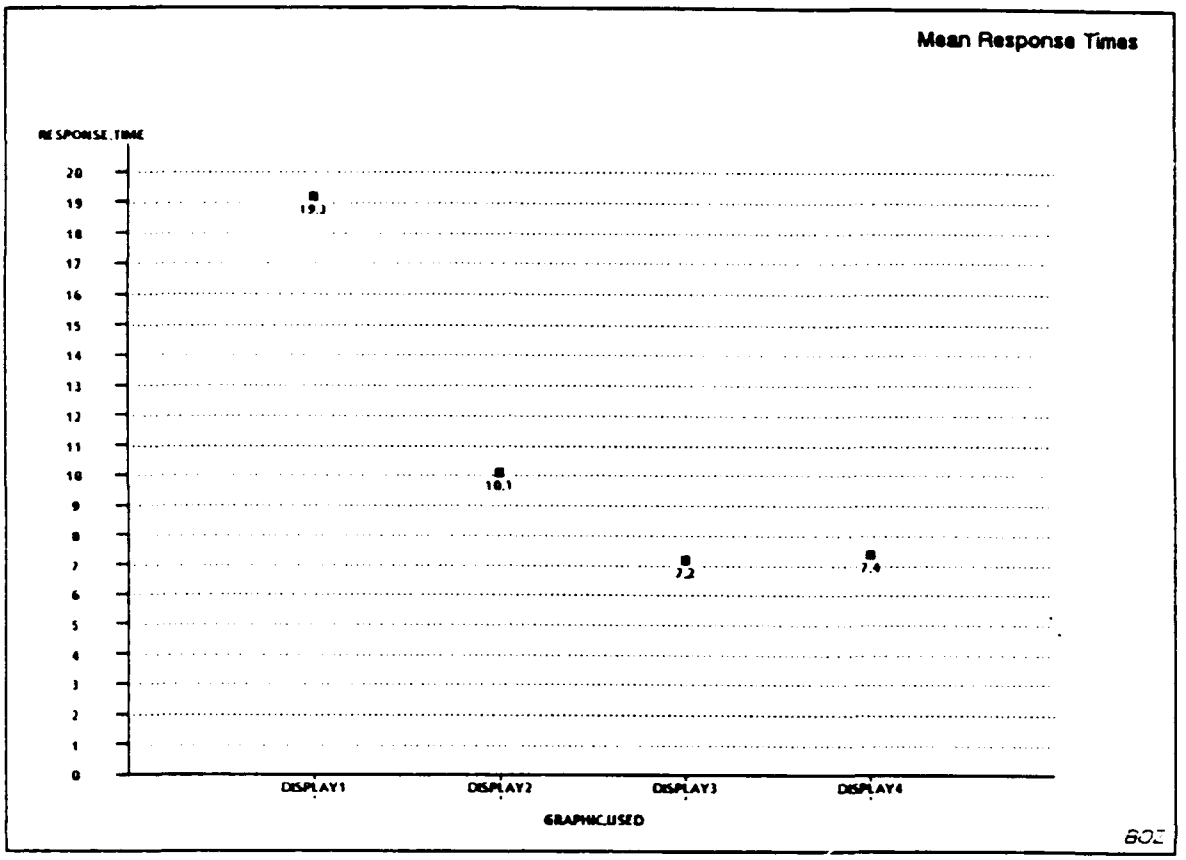


Figure 68: Participants' Mean Performance Times for the Airline Reservation Task.

Computation and search reductions. The next step was to try to understand how participants obtained the observed efficiency advantages, that is, to understand how the time savings could be attributed to the advantages of the presentations that we hypothesized. The first step in this analysis was to attempt to identify which search strategy participants followed for each presentation. Five regression models were created, one for each of rowSearch, rightOfSearch, rightOfSearchU, and two variations on the rowSearch procedure in which the order of the operators was rearranged. These two additional procedures are referred to here as closeSearch and cheapSearch. closeSearch proceeds by considering first those pairs of flights that appear closest together in the presentation. cheapSearch proceeds by considering the shortest flight boxes first.

Only the three parallelizations of the rowSearch procedure provided reasonable fits to the data (closeSearch and cheapSearch did not). Differences in goodness of fit between the three parallelization models were not significant, preventing conclusions to be drawn about which procedures participants applied to which presentations.

Based on these results, the following process was used to assess the effect of substitution of perceptual for logical operators. For each display the most efficient rowSearch procedure supported by that presentation was assumed (i.e., rowSearch for the tabular presentation, rightOfSearch for Presentation 2, and rightOfSearchU for Presentations 3 and 4). It was assumed that the time for one search step was the same in all presentations (although the number of such steps varied with the search procedure supported).

A regression of response times to the number of search steps and cost and layover computations produced a well-fitting statistical model with $F(4, 238) = 73.108$, $p < .0001$, $R^2 = .48$. Removing from the model the counts for either search or checking layovers dramatically reduced the fit. In contrast, removing the counts for checking costs had no effect on the fit. This model yielded the following parameter estimates:

- One search step requires 330 ± 35 milliseconds.

- The `determine-horz-distance` operator is $2 \pm .25$ seconds faster than the `subtractTimes` operator.

- The `judge-heights` operator is negligibly 100 to 300 milliseconds slower than the `addCosts` operator.

These results are consistent with the global time differences given above. The reduced performance time with successive presentations arises for two reasons. First, attending

only to boxes to the right of the current box and to unshaded boxes reduces the number of items that must be searched. Second, substitution of `determine-horz-distance` for `subtractTimes` produces a substantial savings in time. In contrast, the `judge-heights` provided no such advantage. This may have occurred since it is the one perceptual operator that requires integrating quantitative estimates from two different spatial locations. These two effects are sufficient to account for the response time differences between Presentations 1 through 3, and the lack of difference between Presentations 3 and 4.

We can hypothesize that differences in standard deviation between presentations suggest that participants were able to more quickly acquire skill at using the more cognitively efficient solution strategies.

The fact that the regression models did not provide strong enough fits to pinpoint which search strategy participants followed suggests that participants did not all visit the same items in the same order when performing the task. There are many subtle variations on each strategy in which participants may adjust the order in which they visit the items in the display. For example, when using Presentation 4, participants may have searched the left-hand side of the presentation from top to bottom and the right-hand side from bottom to top. This minor variation would cause participants to find the solution earlier or later than would the standard rowSearch procedure. Such variations in search strategy may not necessarily be intentional. Performing the more complex strategies such as rowSearch requires the use of a fairly sophisticated pointer-keeping technique so that users may keep their place in the presentation, not visiting a single item more than once. It may have been that participants intended to follow a version of the rowSearch procedure but were unable to do so. Alternatively, some participants may have completely abandoned the rowSearch procedures and followed some other strategy in which they skipped around in the presentation randomly until they found a solution. This effect was found by van Nes et al (1987) in a similar study of graphic presentation use. van Nes et al were able to generalize

two groups based on search strategy followed: "readers" and "hoppers." Readers scanned the display in the manner described by the rowSearch procedure. Hoppers followed the more random procedure by skipping around in the presentation trying to locate the solution more quickly by chance. The van Nes et al data introduce another possibility as well. It seems that users must spend some amount of time visiting locations within the display where *no* items are present to verify the fact that no items appear there. The time spent doing this seems to vary between participants and may be a function of the visual processing skill of the individual user.

6.4 Conclusion

The results suggest that participants' performance with the BOZ-designed graphic presentations reflect the cognitive efficiency advantages that lie at the core of BOZ's design approach. Further, it is important to note that the theoretical predictions obtained using the simulation tool described in Chapter 4 fairly well accounted for users' performance in advance. This suggests that the methods for generating theoretical predictions could at some point be used in practical situations in lieu of experimentation with real users. One important limitation seems to presently disallow the use of theoretical predictions alone as a means of reliably demonstrating effectiveness. BOZ's ranking system for perceptual operators will require more empirical observation before it is able to make reliable predictions. The experiment reported here showed that BOZ hypothesized that the `judge-heights` perceptual operator would be performed more efficiently than mental addition. This assumption proved to be false in practice. It is likely that future experiments that test graphics that use many different graphical conventions will uncover other problems with BOZ's operator ranking approach.

CHAPTER 7

CONCLUSIONS AND DISCUSSION

7.1 Summary

The research described above adopts a task-analytic approach to the design of graphic presentations that are customized to the requirements of specific tasks. The important distinction made in a task-analytic design methodology is that the effectiveness of a graphic presentation depends on choosing the right graphic for the right task. That is, the utility of a presentation is linked to the nature of the task to be supported more than the information to be represented. An important goal was to create a design theory that was detailed enough to relate hypothesized advantages of graphic presentation-based task performance to particular graphic design features. The task-related advantages of graphic presentations were used to articulate a technique for transforming descriptions of logical procedures and facts into informationally equivalent perceptual procedures and graphic presentations. The task-analytic design approach was also used to create measures of cognitive complexity that allow logical and alternative perceptual procedures to be compared. The implementation of the design theory, BOZ, was used to design perceptual procedures and graphic presentations to support performance of a set of real-world tasks. An empirical study was conducted to determine the extent to which the advantages of one BOZ-designed presentation were reflected in participants' performance of the target task.

7.2 Contributions

Design of Effective Graphic Presentations. What are the real graphic design successes of BOZ? Has BOZ produced a graphic that no one else has designed before? Given its current set of perceptual operators and graphical presentation objects it is unlikely

that BOZ will produce new graphic designs that differ radically from existing designs. That is, in the examples developed in this dissertation, no graphic seems to contain a shape or a configuration of shapes that strikes the reader as being completely novel, having never appeared in a graphic used in one domain or another. This comes as little surprise as the study of novel spatial elements and their mathematical properties enjoys centuries of prior investigation. Rather, the contribution of the present version of BOZ lies in forming compositions of existing designs to arrive at presentations that gather the relevant features of several individual designs to arrive at "customized" presentations that support the operators in a target user task. It is important to note that the degree of customization achieved by each presentation is more than a simple collection of useful presentations implicated by a task description. Rather, BOZ is additionally able to assign degrees of importance to the individual operators in a task and use this information to make sacrifices in less important aspects of a task to facilitate efficient perceptual processing in more demanding aspects of the same task.

A second graphic design contribution is the technique for obtaining quantitative predictions about the relative utilities of a set of alternative presentations with respect to a target task. These measures may eventually allow designers to assess the effectiveness of a presentation design without performing costly empirical evaluations. The present design evaluation model fails to predict two important task performance measures, namely, performance times for the perceptual operators in BOZ's catalog and measures of individual differences in skill among users. These limitations are discussed in Section 7.3.

A Formalism for Describing Perceptual Inference Procedures. A second contribution of the research is to advance a working formalism for describing perceptual inference procedures. The formalism is used to capture perceptually executed procedures using the same framework used to describe abstract procedures. Having a formal representation for perceptual inference procedures also allows us to make detailed

comparisons between alternative procedures, within the same modality as well as across modalities.

An important obstacle to constructing any descriptive model of perceptual inferencing is that our understanding of how humans perform many primitive perceptual tasks is largely incomplete [Ullman, 1984]. The formalism proposed here circumvents this obstacle by treating perceptual operators as black box functions that describe perceptual inferences only in terms of the inputs they receive and the outputs they produce. The price paid in making this assumption is that it is in general impossible to understand what low-level perceptual procedures users are following when performing a task. Chapter 6 showed how this prevents us from drawing strong conclusions from experimental data collected from users during task performance.

The perceptual task formalism is organized around the set of primitive graphical languages listed in Appendix 1. Each primitive graphical language contains the following items: (a) the perceptual operators associated with each language; (b) the graphical objects that are manipulated by the perceptual operators; and (c) the general form for expressing graphical facts using the graphical language. The formalism can be used to create working descriptions of arbitrary perceptual procedures. Perceptual procedures are ordered sequences of calls to the set of perceptual operators. Input data to a perceptual procedure are defined using the general forms for graphical facts. Once a perceptual procedure and set of graphical facts has been created they can be executed using BOZ's simulation tool. Section 7.3 discusses several limitations of the perceptual procedures that are *automatically* generated from logical procedures submitted to BOZ.

Automated Graphical Presentation. Aside from an algorithmic specification of a design theory for graphic presentations, BOZ appears potentially useful as a tool for the automated design and generation of graphic presentations in computer information systems.

However, two limitations of the present version prevent BOZ's current use in real-time applications: (a) the need to hand-generate the task descriptions required by BOZ as input; and (b) the run time of BOZ's current implementation. Both of these issues are discussed in detail in Section 7.3.2.

7.3 Limitations of BOZ

7.3.1 Cognitive Issues

Learning graphical conventions and procedures. An important aspect of the utility of a graphic presentation not directly addressed in the task-based graphic design theory is the time required to understand the procedures that must be followed to use that presentation successfully, and to acquire the skills necessary to perform them.[1] There seem to be no inherent advantages of learning a perceptual procedure instead of a logical one even if that perceptual procedure is eventually performed more efficiently. The learning issue is relevant when graphics are presented in "walk up and use" situations where it is unsafe to assume prior knowledge or skill on the part of the user. Learning issues are less important in skilled performance task situations. Three features of BOZ raise important issues that pertain to learning graphical conventions and procedures.

First, the very idea of presenting users with different presentations customized to different tasks by definition increases the number of new graphic conventions that the user must learn and raises important issues related to user interface consistency [Shneiderman, 1985]. Since choices made about how to allow users to perceptually obtain a result during a task critically depend on the other requirements of that task, it is routinely the case that the user must perform different perceptual operators to obtain the same result in different tasks. For example, the airline graphic in Figure 24 requires the user to search for an appropriately labelled rectangle to locate a flight originating in a given city. On the other hand, the airline

[1] Weintraub was first to make the point that the visual procedures required to use a graphic have to be learned and that the time required for this learning is a significant graphic design parameter.

graphic in Figure 36 requires the user to search the horizontal axis to locate this same information. Since the conventions are inconsistent between the two presentations, arguments about failing to exploit procedural transfer or even causing negative transfer can be made [Kieras and Polson, 1985]. This issue reduces to the following question. *Do the benefits of presentation customization outweigh the increase in learning time due to inconsistency?* The answer to this question of course is unique to the specific goals of each task situation (e.g., the utility of computers seems to generally outweigh the two years required to learn to program them but this depends on the application in which they are to be deployed). The answer to this question also depends on having a sense of the value of consistency, a measure upon which two investigators seldom agree [Grudin, 1990].

Second, one strategy used by BOZ for minimizing the learning time for a set of graphical conventions is to maintain *one-to-one correspondences* between primitive graphical languages and relational domain sets. Graphics in popular use routinely violate the one-to-one correspondences principle in two ways. First, graphics sometimes use two or more perceptual dimensions to redundantly encode a single dimension of information. For example, some bar charts redundantly represent quantities with both the height of the bars as well as shading or color. The user of the bar chart can either judge the height of the bar or determine its color to arrive at a quantity. In these cases it is assumed that allowing the reader a choice between several perceptual procedures for extracting a single datum or obtaining a single result has benefits in that the presentation may be more flexible or allow the reader to corroborate the inferences they draw using separate procedures. Kosslyn (1989) and Ohlsson (1987) argue the opposite case: that allowing redundant encodings only serves to confuse the reader when mapping perceptual dimensions onto the dimensions of information being displayed. Tufte (1983) presents a similar argument relating learning complexity to the amount of "data ink" appearing in a graphic. Second, many graphics use extraneous perceptual dimensions as ornamentation to make graphics look more appealing. Tufte (1983) criticizes this practice claiming that graphical

decorations complicate learning. Arguments made against the use of redundant or decorative graphical dimensions assume (perhaps correctly) that reader approaches the learning problem using a *forward search* strategy, first collecting the perceptual dimensions appearing in a graphic, and then mapping them to the data dimensions being conveyed. A *backward search* strategy proceeds by first noting the data dimensions to be conveyed and then locating the corresponding perceptual dimensions that encode them. It is straightforward to show that redundant encodings *increase* learning time when forward search is used and *decrease* learning time when backward search is used. What search strategies users actually follow along with the status of redundant encodings is an open question, with no particular argument yet supported by data.

Third, a second way in which BOZ tries to minimize learning time is that when choosing graphical presentation objects, BOZ always selects the simplest object (of <point>, <line>, <rectangle>, and <polygon>) that is sufficient to encode all of the relations appearing in the perceptual data structure for that object. Tufte (1983) argues that learning time is minimized when a graphic uses the minimum amount of "data ink" to encode a data set. For example, plot charts and bar charts are popularly used to display a set of values to support simple comparison tasks. Tufte argues that the plot chart is the better presentation since it encodes the same information as the bar chart using less data ink. The data ink argument is rooted in the observation that the graphical objects in the bar chart also have width, area, and shading which may entice the reader into thinking that they are meaningful. One qualification of this point is necessary. All graphic presentations necessarily use all of the graphical dimensions (primitive graphical languages). For example, the points in the plot chart indeed have a color, height, width, area, slope, connectivity, etc. One can argue the case that these dimensions may also distract the reader's attention. Thagard (1989) distinguishes these two cases by arguing that the reader's attention will be attracted only when the values of these dimensions vary

unnecessarily within a presentation. That is, the reader will attend to the color of a set of points in a plot chart only when the color of the points varies unnecessarily.

Two further issues are relevant to the task of learning a new perceptual procedure. First, many users have previous experience using graphics that not only improve their performance of specific perceptual operators, but also provide practice in assigning interpretations to them. That is, the process of learning a perceptual procedure that manipulates information about time along a scaled horizontal axis is facilitated by practice that users have had with other graphics that use that same convention. Second, many graphic presentations exploit learned real-world conventions and metaphors. For example, a graphic presentation of a set of politicians and their relative political stances could exploit the "right/left" convention used in everyday conversation, conservative politicians being placed on the right side of a horizontal axis, liberals to the left.

Descriptiveness of BOZ's Perceptual Procedures. Perceptual procedures are produced by BOZ using the simple strategy of substituting the names of the selected perceptual operators in place of the names of the original logical operators in a logical procedure. This raises an important question about the amount of descriptiveness captured in the perceptual procedures produced by BOZ. This question can be asked at two levels: (1) about the perceptual operators themselves; and (2) about the perceptual procedures defined using the perceptual operators.

At the perceptual operator level, BOZ has no characterization of human perceptual processing beyond the logical specifications that comprise the perceptual operator definitions. That is, perceptual operators are nothing more than logical inferences that have been relabeled with suggestive names. This is the notion of a mathematical renaming, the basis for the notion of *substitution* given in Definition 1, and the core assumption operationalizing BOZ's approach. Because of this "black box" approach, perceptual

operators (and the perceptual procedures constructed using them) do not formally capture the representation-dependent context in which inferences are performed. An example of this is the difference between search for a fact in a factbase and search for a point in the plane. The perceptual operator does not contain procedural descriptions of scanning the eye across a horizontal axis, recognizing a label, running one's finger up the column defined by the horizontal position, and scanning for the desired object.

At the perceptual procedural level, in addition to propagating the uncertainties of each individual POP appearing in a procedure, BOZ's perceptual procedure derivation technique is not able to create alternative perceptual procedures defined over a single graphic presentation beyond what is derivable through application of the operator parallelization rules. This is especially important when alternative strategies derived through operator reordering are likely to be followed by users. The graphic in Figure 36 was an example for which users are likely to follow a procedure that differs from the one generated by BOZ by way of the order in which the operators are applied. It was shown that alternative procedures derived through operator reordering are of equal efficiency in general. What is potentially lost however is an accuracy to which the BOZ-generated perceptual procedures describe the steps that users actually follow.

Two basic points defend BOZ's operator substitution approach. First, the descriptiveness of any perceptual operator is upper-bounded by the depth of our understanding of how humans actually perform primitive perceptual tasks. Ullman (1984) reviews work in this area and concludes that the details of primitive perceptual task performance are largely undiscovered, providing many examples of simple perceptual tasks, such as inclusion judgements, for which there is no agreement upon even the overall strategies used by humans. In the absence of details about primitive tasks, investigations of graphic utility for complex information-processing tasks may well best be described as "playing twenty

questions with nature [Newell, 1973]." Strong assumptions about how perceptual tasks are performed may provide little leverage in predicting presentation utility.

Second, the problem of deriving alternative, more efficient procedures from existing procedures is an instance of the more general problem of analysis of algorithms known to be not computable by algorithm in general. We can adjust the question by asking whether or not we can write a list of transformation rules that capture known procedural "shortcuts" that allow the same graphic to be used in a different way to obtain the same result. While no such list of rules presently exists it seems promising that future studies of perceptual task performance may begin to catalog them.

Social and ethnographic factors in graphic presentation use. Other works have investigated social factors involved in graphics use. Understanding the conventions used in a novel presentation appears to be highly sensitive to culture [Hudson, 1968; Garfinkel, 1972; Cahill, 1976]. That is, different groups may be more pre-disposed to assign meanings to a set of graphical signs and symbols other than the meanings chosen by BOZ. Also, perceptual task performance itself may differ among groups since different groups may have skill or experience using graphical conventions different from those used by the populations of users studied here (or in using none at all). In this case, the set of perceptual operator rankings used by BOZ would inappropriately describe what is performed most efficiently by these groups. This evidence suggests that social factors influencing the learning and performance of perceptual procedures lies outside the scope of available information-processing models of cognition and certainly outside the scope of any formal theory of graphical syntax and semantics. Future investigations of the pragmatic, cultural, and social aspects of graphics use are likely to prove at least as useful as results that address syntax and semantics.

Performance time predictions. Another limitation of BOZ is that without exact quantitative estimates of the difficulty of logical and perceptual operators, BOZ is unable to make predictions about the time required to complete a perceptual procedure using a graphic presentation. Instead, BOZ produces a ranked set of alternative perceptual procedures that accomplish a stated goal. Two special circumstances seem to prevent BOZ from achieving reliable zero-parameter performance time predictions. First, unless the complexity of primitive logical and perceptual operators were completely independent of presentation, task, and context, the wide variety of uses of presentations would seem to disallow making general predictions. Second, other investigators have shown that the level of perceptual skill of the user greatly affects important measures of their performance. Hence, BOZ's predictions could at best predict mean performance times for large populations [Kieras and Poison, 1985]. As variance between tasks and users become large, the meaningfulness of these predictions decreases.

7.3.2 Automated Graphic Design Issues

Hand-Generated Task Descriptions. The descriptions of logical procedures required by BOZ as input must presently be hand-generated. This prevents BOZ from being used directly by end users. A future research topic is to investigate ways of automatically generating task descriptions, eliminating the need for human intervention. SAGE [Roth et al, 1989] uses a discourse processor that allows descriptions of simple operators to be generated by analyzing simple natural language queries made by the user. However, this approach is unable to generate descriptions of complex procedures defined using collections of many operators. An alternative approach to lowering the boundaries to task specification is to generate task descriptions from user queries expressed using a restricted database/factbase query language. A Prolog-like query language has been partially implemented that allows the user to type queries into a dialogue window. These queries are then compiled into the standard LOP notation used by BOZ. An example query using this language is shown in Figure 69.

```
flight(F):-departure(F) > 9:00, arrival(F) < 18:00,
           cost(F) < 300, availability(F) = ok.
```

Figure 69: A Prolog-like Factbase Query Language

BOZ's Run Time. While the run time complexity of BOZ may theoretically be able to meet the demands of on-line information systems, the present implementation fails to produce graphics in a time that would be considered acceptable by computer users. The rendering component is particularly slow for graphics containing many graphical objects. The search complexity for BOZ's perceptual operator substitution component is presently: $T_{\text{operator substitution}} = n * c * t$, where n is the number of logical operators appearing in a procedure, c is the number of possible operator classes that each logical operator must be matched against, and t, the time required to find all matches for a single operator. The Xerox 1186 implementation of BOZ required about twelve seconds to classify the airline reservation task operators (14 operators matched against 11 operator classes) shown in Figure 14. BOZ's perceptual data structuring algorithm is linear in the number of logical operators, n, and domain sets, d: $T_{\text{data structuring}} = n * d$. BOZ required less than one second to design the initial perceptual data structure shown in Figure 18. The perceptual operator selection component runs $nlog(n)$ in the number of logical operators: $T_{\text{operator selection}} = n * (n - s)$, where s is the number of operators that have been selected thus far. The perceptual operator selection component required six seconds to select the perceptual procedure and data structure shown in Figures 19 and 20. The rendering component is linear in the number of structured facts to be presented, f, and the number of primitive graphical languages, p, appearing in each structured fact: $T_{\text{rendering}} = f * p$. The rendering component required approximately eighteen seconds (including logical to graphical fact translation) to render the flights presentation in Figure 24 and approximately one minute fifteen seconds to generate the seating chart presentation in Figure 25.

Overall, BOZ designed both presentations in about nineteen seconds, rendering the flights and seating charts presentations after thirty-seven seconds and one minute forty-five seconds, respectively. BOZ's current run time does not fall within an acceptable standard for real-time data presentation. A future research topic is to investigate more efficient implementations of BOZ.

7.4 Other Advantages of BOZ-Designed Graphic Presentations

Working memory advantages. Limitations on the capacity of short-term memory introduce an everyday feature of performance of complex information-processing tasks: that items are often lost during problem solving when the number of items that must held temporarily in memory exceeds a given limit. Graphic presentations appear to help alleviate this problem in three ways. First, an additional advantage of step skipping (see Chapter 1) is that eliminating operators from procedures relieves the user from having to maintain the results of those operators in working memory. For example, for the airline reservation task, when determining the layover between two flights using logical or tabular representations, it is necessary to first look up the departure and arrival times, maintain them in working memory, and then perform the subtraction. At this point the two individual times can be replaced with the single layover time. Using the graphic presentation, the look up steps may be skipped. That is, the user can simply determine the layover and store this in working memory. Second, perceptual data structuring often groups multiple facts in a single perceptual chunk. This allows users to store a single perceptual chunk in place of a set of individual facts. Third, when items are lost from working memory, the efficiency at which many perceptual operators are performed suggests that graphic presentations may allow users to more quickly retrieve or recompute lost items.

Efficient Referencing. Graphic presentations also facilitate efficient referencing. When used for communication, graphics allow users to substitute pointing in place of more

elaborate verbal reference-securing. Football diagrams allow players to quickly reference individual players, formations, and paths of motion. The use of pronouns and adjectives such as it, him, this, here, etc. indicate that "chalktalk" sessions between players and coaches rely heavily on the use of pointing for reference-securing.

7.5 Advantages of Graphic Presentations Not Addressed by BOZ

Control knowledge. Larkin (1989) points out that another advantage of graphic presentations is that they sometimes encode information about the state of a problem solution as well as indications to the next step that must be performed. Consider the problem of determining the number of possible paths from one point in a city to another. If we represent the problem as a graph, we can quickly obtain the answer in the following way. If we number the node corresponding to the starting point 0, we can number each successive node so that it is the sum of the two node previous nodes leading back to the starting point as shown in Figure 70.

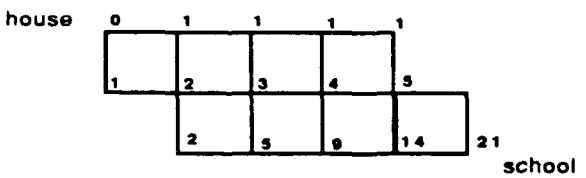


Figure 70: A Graphic Presentation Containing Control Information

If we number all of the nodes in the graph in this fashion the number placed on the final node (the endpoint) is the number of unique paths from the starting point to the ending point. At each step in the problem the next step in the solution is easily identified as labeling that node(s) that is both unlabeled and exactly one node away from the node last labeled. In this way control information about the algorithm is directly encoded in the problem representation. Koedinger (1989) describes a similar protocol example in which a geometry expert began to place numbers on sides of a geometric figure. The numbers were used to make rapid inference about proportional relationships between the sides of the

figure. The process of "filling in the blanks" on the geometry figure led the expert to an immediate solution after struggling with the problem for 45 minutes. Larkin and Simon (1987) make the same point in showing how application of problem-solving operators is guided by the ropes connecting a set of pulleys in a mechanics problem.

Despite the apparent encoding of control information in the examples and their utility to the problem solver, it is not clear that such control schemes follow from first design principles and can be replicated in arbitrary situations, or are epiphenoma of other representation features.

Improving Recall for Data. Graphics also appear to help users recall presented data. In fact, the memorability of graphic presentations was William Playfair's main argument for using them. Experimentalists such as Washburne (1927) have shown graphics to be useful along this dimension but not in an unqualified sense. Washburne showed that the way data is laid out in a particular graphic design can effect users' ability to recall that data. Figure 71 shows two bar charts that differ only in the arrangement of the bars. The bar chart to the right is more likely to be remembered since the stairwise arrangement of the quantities allows users to encode them in a single perceptual chunk. That is, users can reduce their memorization task to two items: the order of the labels along the horizontal axis (i.e., d a c b), and the incremental difference between the quantities.

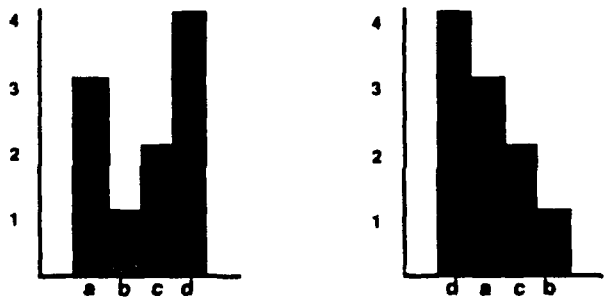


Figure 71: A Memorable Graphic Presentation.

Since BOZ does not manipulate memory for data as one of its theoretical parameters, it presently has no capability for arranging data in ways known to improve recall.

7.5 Further Issues

Extended Model of Mathematical Metaphor. The number of different kinds of presentations that BOZ is able to produce is limited by the perceptual operators in the perceptual operator catalog and the set of corresponding graphical presentation objects. BOZ's present catalog of perceptual operators contains operators to support simple visual search, and perceptual arithmetic and logical computations. This limited set hardly exhausts the set of possible inferences that can be made perceptually. BOZ's current set of operators and objects is limited in the following way. Many graphical objects and shapes, such as the graphs of many studied functions (e.g., circle, parabola, tangent) enjoy special computational properties that can be used to derive geometric solutions to problems that are equivalent to propositional solutions of the same problems. For example, if we can take the derivative of a function graph at any point by drawing a line tangent to the graph at that point and judging the slope of the tangent line. Similarly, we can graphically support the task of computing values of the function πr^2 by allowing the user to draw circles on the screen and then unwrapping the figure into a straight line so it can be measured. Incorporating these features in BOZ would require a more sophisticated semantic theory for the graphical presentation objects in BOZ's repertoire. BOZ presently contains only the <point>, <line>, <polygon>, and <rectangle> objects and information about relationships between their endpoints and planar dimensions.

How Much Information Can Be Encoded in a Graphic? Another interesting question raised by BOZ is that of how much information can be encoded in a single graphic. This question decomposes into two questions: (1) How many different dimensions of information can be encoded in a graphic? and (2) How many values can be encoded within each dimension? In the information-theoretic sense, the amount of

information encodable in a single presentation is upper-bounded by the number of available primitive graphical languages, assuming that each graphical language is used to encode exactly one domain set of information. Of course, this bound is further restricted by the processing capabilities of the user. Bertin (1983) proposes that a maximum of three dimensions of information can be accommodated in a single presentation, each dimension using a single graphical language. Bertin bases his argument on the observation that the human user is generally unable to perceive and maintain more than three perceptual dimensions in short-term memory at any one time. It is interesting to note that the graphic in Figure 61 encodes *six* dimensions of information.

The second part of this question concerns the number of data items that can be effectively presented using any one graphic design. Any graphic design has a practical limit on the number of items that can be accommodated in the plane of the graphic. For instance, the graphic airline schedule in Figure 24 is likely to accommodate a maximum of twenty or thirty flights. Displaying more flights in the graphic would require that we decrease the size of each flight box to a size that is not easily or accurately perceptible. Similarly, adding more connections to the airline graphic in Figure 47 would greatly complicate the user's task of tracing the connecting lines through the presentation.

7.6 Concluding Remarks

This research was an attempt to combine two worlds. It was an attempt to collect the design problems and constraints of two worlds of people who design information artifacts: Computer Scientists and Applied Cognitive Psychologists. In the real world of design practitioners of the computer and psychological sciences operate independently. Artifacts are subjected to a gauntlet of design criteria, each world of designers working their magic and passing the artifact along to the next. Artifacts are designed by those who understand run time efficiency, implementation costs, and high functionality, and are then redesigned as well as can be done by those sympathetic to the issues of human factors. It has been the

goal of this dissertation to show that another strategy exists: a strategy in which both worlds work on the artifact at the same time using a set of interdisciplinary tools. BOZ is an attempt to create a computational model of design that includes the design constraints of both worlds in a single set of design elements that inherit the constraints of both design worlds. The perceptual operators and graphical objects that comprise BOZ's design elements are computational constructs that have specific psychological implications. In BOZ there is no way to choose a perceptual operator or a graphical object without thinking about both sets of design issues. For each perceptual operator and graphical object, BOZ makes explicit its computational power as well as the degree of success users are likely to achieve when using it. BOZ's interdisciplinary approach does not purport to eliminate conflicts between design constraints. Rather, designers from both worlds choose among BOZ's elements together. This allows the consequences of every design choice for either world to be made explicit at each step in the design process. It is hoped that providing a context in which designers from different worlds can work together (or at the same time), artifacts that are both highly functional and highly usable can be more consistently designed. As more is understood about the way humans interact with information artifacts, implicating other disciplines, a future research challenge will be to investigate ways of further collapsing the design of information artifacts into a shared enterprise.

APPENDIX

APPENDIX

BOZ'S PRIMITIVE GRAPHICAL LANGUAGES

The following describes the complete set of primitive graphical languages and their associated perceptual operators. Each entry contains the following elements:

(a) The set of data types that can appear in graphical facts and perceptual operators associated with the primitive graphical language.

(b) The expressiveness of the primitive graphical language.

(c) The notation for expressing graphical facts in the primitive graphical language.

(d) The perceptual operators (POPs) associated with the primitive graphical language.

HorzPos

Graphical Language Data Types

$\mathbf{P}_{\text{HorzPos}}$ = {<point> <line> <rectangle> <polygon> <label> <horzpos> <real> <integer>}

Domain Set Expressiveness

Nominal = 25

Ordinal = 25

Quantitative = 500

Graphical Fact Notation

(HorzPos <object> <horzpos>)

where <object> := {<point> <line> <rectangle> <polygon> <label>}

Perceptual Operators (POPs):

SEARCH OPERATORS
- determine-horz-pos
- search-obj-at-horz-pos
- search-any-horz-obj
- verify-obj-at-horz-pos

COMPUTATION OPERATORS
- left-of?
- right-of?
- horz-coincidence?
- determine-horz-distance
- horz-forward-projection1
- horz-forward-projection2
- horz-multiple-projection
- horz-forward-projection1
- horz-forward-projection2
- find-horz-midpoint
- find-horz-midpoint

VertPos

Analogous to the HorzPos language.

Shading

Graphical Language Elements:

$\mathbf{P}_{\text{Shading}}$ = {<rectangle> <polygon> <shade>}

Domain Set Expressiveness:

Nominal = 3

Ordinal = 3

Quantitative = 0

Graphical Fact Notation

(Shading <object> <shade>)

where <object> := {<rectangle> <polygon>}

Perceptual Operators (POPs):

SEARCH OPERATORS
- determine-shade
- search-obj-with-shade
- search-any-obj-and-shade
- verify-obj-and-shade

COMPUTATION OPERATORS
- lighter-shade?
- darker-shade?
- same-shade?

Shape

Graphical Language Elements

$\mathbf{P}_{\text{Shape}}$ = {<polygon> <shape>}

Domain Set Expressiveness

Nominal = 4

Ordinal = 0

Quantitative = 0

Graphical Fact Notation

(Shape <object> <shape>)

where <object> := {<polygon>}

Perceptual Operators (POPs):

SEARCH OPERATORS

determine-shape
search-obj-with-shape
search-obj-and-shape
verify-obj-and-shape

COMPUTATION OPERATORS

same-shape?

Height

Graphical Language Elements

$\mathbf{P}_{\text{Height}}$ = {<rectangle> <height> <integer> <real>}

Domain Set Expressiveness:

Nominal = 5

Ordinal = 10

Quantitative = 500

Graphical Fact Notation

(Height <object> <height>)

where <object> := {<rectangle>}

Perceptual Operators (POPs):

SEARCH OPERATORS
- determine-height
- verify-obj-and-height
- search-obj-with-height
- search-obj-and-height

COMPUTATION OPERATORS
- taller?
- shorter?
- same-height?
- stack-heights1
- stack-heights2
- stack-heights3
- height-difference

Width

analogous to Height

Line Length

Graphical Language Elements

$\mathbf{P}_{\text{LineLength}}$ = {<line> <linelength> <integer> <real>}

Domain Set Expressiveness

Nominal = 500

Ordinal = 10

Quantitative = 5

Graphical Fact Notation

(LineLength <object> <linelength>)

where <object> := {<line>}

Perceptual Operators (POPs)

SEARCH OPERATORS
- determine-line-length
- verify-line-and-length
- search-line-with-length
- search-line-and-length

COMPARISON OPERATORS
- line-longer?
- line-shorter?
- same-length?
- length-projection1
- length-projection2
- multiple-length-projection
- length-difference

Area

Graphical Language Elements

$\mathbf{P}_{Area}$ = {<rectangle> <area> <integer> <real>}

Domain Set Expressiveness:

Nominal = 5

Ordinal = 10

Quantitative = 500

Graphical Fact Notation

```
(Area <object> <area>)
   where <object> := {<rectangle>}
```

Perceptual Operators (POPs):

SEARCH OPERATORS

```
determine-area
verify-obj-and-area
search-obj-with-area
search-obj-and-area
```

COMPUTATION OPERATORS

```
bigger?
smaller?
same-area?
```

Line Dashing

Graphical Language Elements:

$\mathbf{P}_{LineDashing}$ = {<line> <rectangle> <polygon> <linedashing>}

Domain Set Expressiveness

Nominal = 2

Ordinal = 2

Quantitative = 0

Graphical Fact Notation

```
(LineDashing <object> <linedashing>)
   where <object> := {<line> <rectangle>}
```

Perceptual Operators (POPs):

SEARCH OPERATORS
- determine-line-dashing
- search-line-with-dashing
- search-line-and-dashing
- verify-line-and-dashing

Line Thickness

Graphical Language Elements:

$\mathbf{P}_{\text{LineThickness}}$ = {<line> <rectangle> <polygon> <linethickness>}

Domain Set Expressiveness:

Nominal = 4

Ordinal = 5

Quantitative = 4

Graphical Fact Notation

(LineThickness <object> <linethickness>)

where <object> := {<line> <rectangle> <polygon>}

Perceptual Operators (POPs):

SEARCH OPERATORS
- determine-line-thickness
- verify-line-and-thickness
- search-line-with-thickness
- search-line-and-thickness

COMPUTATION OPERATORS
- line-thicker?
- line-thinner?
- same-thickness?
- thickness-projection
- multiple-thickness-projection
- thickness-difference

Connectivity

Graphical Language Elements:

$\mathbf{P}_{\text{Connectivity}}$ = {<point> <label> <rectangle> <polygon>}

Domain Set Expressiveness:

Nominal = 10

Ordinal = 0

Quantitative = 0

Graphical Fact Notation

```
(Connectivity <object> <object>)
   where <object> := {<point> <rectangle> <polygon> <label>}
```

Perceptual Operators (POPs):

SEARCH OPERATORS
- object-connected-to-another?
- find-connectee
- connected?

Labels

Graphical Language Elements:

$\mathbf{P}_{\text{Labels}}$ = {<point> <line> <rectang ≥> <polygon> <label>}

Domain Set Expressiveness:

Nominal = ∞

Ordinal = ∞

Quantitative = ∞

Graphical Fact Notation

```
(Labels <object> <label>)
   where <object> := {<point> <line> <rectangle> <polygon>}
```

Perceptual Operators (POPs):

SEARCH OPERATORS
- read-label
- search-object-with-label
- verify-object-and-label
- search-any-object-and-label

COMPUTATION OPERATORS
- same-label?
- greater-than-label?
- less-than-label?
- add-labels
- subtract-labels
- multiply-labels
- divide-labels

Visibility

Graphical Language Elements:

$\mathbf{P}_{\text{Visibility}}$ = {<point> <line> <rectangle> <polygon> <label> <visibility>}

Domain Set Expressiveness:

Nominal = 2

Ordinal = 2

Quantitative = 0

Graphical Fact Notation

```
(Visibility <object> <visibility>)
   where <object> := {<point> <line> <rectangle> <polygon> <label>}
```

Perceptual Operators (POPs):

SEARCH OPERATORS

```
search-any-visible-object
verify-object-visible?
search-visible-object
```

BIBLIOGRAPHY

BIBLIOGRAPHY

Albers, J. (1963). *Interaction of Color.* New Haven, 1963.

Alden, D.G., Wedell, J., and Kanarick, A.F. (1971). Redundant stimulus coding and keeping-track performance. *Psychonomic Science* **22**, 201-202.

Anderson, J.R. (1983). *The Architecture of Cognition.* Cambridge, MA: Harvard University Press.

Arani, T., Karwan, M.H., and Drury, C.G. (1984). A variable memory model of visual search. *Human Factors* **26** (6), 631-639.

Arens, Y., Miller, L., and Sondheimer, N. (1989). Presentation planning using an integrated knowledge base. In J. Sullivan and S. Tyler (Eds.), *Architectures for Intelligent Interfaces: Elements and Prototypes.* Addison-Wesley, Reading, MA.

Arnheim, R. (1969). *Visual Thinking.* Berkeley, CA: University of California Press.

Asch, S.E. (1956). Studies of independence and submission to group pressure: A minority of one against a unanimous majority. *Psychological Monographs*, 70.

Barnard, P., and Marcel, T. (1987). Representation and understanding in the use of symbols and pictograms. R. Easterby and H. Zwaga (Eds.), *Information Design*, NY: John Wiley & Sons, Ltd., 37-75.

Barsam, H.F., and Simutis, Z.M. (1984). Computer-based graphics for terrain visualization training. *Human Factors* **26** (6), 659-665.

Bartlett, M W., and Smith, L.A. (1973). Design of control and display panels using computer algorithms. *Human Factors* **15** (1), 1-7.

Beck, J., and Ambler, B. (1973). The effects of concentrated and distributed attention on peripheral acuity. *Perception and Psychophysics* **14** (2), 225-230.

Behan, F.L., Behan, R.A., and Wendhausen, H.W. (1972). Color perception underwater. *Human Factors* **14** (1), 41-44.

Benbasat, I., and Dexter, A.S (1986). An investigation of the effectiveness of color and graphical information presentation under varying time constraints. *MIS Quarterly*, March, 59-83.

Benbasat, I., Dexter, A.S., and Todd, P. (1986). An experimental program investigating color-enhanced and graphical information presentation: An integration of the findings. *Communications of the ACM*, Volume 29, Number 11, 1094-1105.

Bertin, J. (1983). *Semiology of graphics*. W. Berg, transl., Madison, WI: University of Wisconsin Press.

Beveridge, M., and Parkins, E. (1987). Visual representation in analogical problem solving. *Memory and Cognition* **15** (3), 230-237.

Binns, B., and Clark, N. (1986). The graphic designer's use of visual syntax. In *Proceedings of the Section on Statistical Graphics of the American Statistical Association.*

Bloomfield, J.R. (1972). Visual search in complex fields: Size differences between target disc and surrounding discs. *Human Factors* **14** (2), 139-148.

Bowman, W.J. (1968). *Graphic Communication.* New York: John Wiley and Sons.

Brachman, R.J. and Schmolze, J.G. (1985). An overview of the KL-ONE knowledge representation system, *Cognitive Science* **9** (2), 171-216.

Brainerd, W.S., and Landweber, L.H. (1974). *Theory of Computation*, New York: John Wiley and Sons.

Brinton, W.C. (1916). Graphic methods of presenting facts. New York: *Engineering Magazine.*

Brinton, W.C. (1939). *Graphic Presentation.* New York: Brinton Associates.

Brown, B., and Monk, T.H. (1975). The effect of target surround and whole background constraint on visual search times. *Human Factors* **17** (1), 81-88.

Bundesen, C. and Pedersen, L.F. (1983). Color segregation and visual search. *Perception and Psychophysics* **33**, 487-493.

Cahill, M.C., and Carter, R.C. (1976). Color code size for searching displays of different density. *Human Factors* **18** (3), 273-280.

Cahill, M. (1976). Design features of graphic symbols varying in interpretability. *Perceptual and Motor Skills* **42**, 647-653.

Card, S.K., Moran, T.P., and Newell, A. (1983). *The Psychology of Human-Computer Interaction*, Hillsdale, NJ: Lawrence Erlbaum Associates.

Carter, R.C. (1982). Visual search with color. *Journal of Experimental Psychology: Human Perception and Performance* **8**, 127-136.

Carter, R.C., and Cahill, M.C. (1979). Regression models of search time for color-coded information displays. *Human Factors* **21**, 293-302.

Chakrapani, T.K., and Ehrenberg, A.S.C. (1976). *Numerical Information Processing.* London: London Business School.

Chambers, J.M., Cleveland, W.S., Kleiner, B., and Tukey, P.A. (1983). *Graphical Methods for Data Analysis.* Belmont, CA: Wadsworth.

Chang, S.-K., Shi, Q.-Y., Yan, C.-W. (1987). Iconic indexing by 2-D strings. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **9** (3), 413-428.

Chang, S.-K., Yan, C.-W., Dimitroff, D.C., Arndt, T. (1988). An intelligent image database system. *IEEE Transactions on Software Engineering* **14** (5), 681-688.

Chang, S.-K., Tauber, M.J., Yu, B., Yu, J.-S. (1989). A visual language compiler. *IEEE Transactions on Software Engineering* **15** (5), 506-525.

Chapanis, A. (1975). *Ethnic Variables in Human Factors Engineering*. Baltimore, MD: Johns Hopkins.

Christ, R.E. (1975). Review and analysis of color coding research for visual displays. *Human Factors* **17**, 562-570.

Clancey, S.M. and Hoyer, W J. (1987). Effects of age and skill on domain-specific visual search. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, Seattle, WA, 398-404.

Cleveland, W.S. (1985). *Elements of Graphing Data*. Monterey, CA: Wadsworth Advanced Books and Software.

Cleveland, W.S. and McGill, R. (1984). Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistics Association, 79* (387), 531-554.

Cuff, D.J. (1973). Colour on temperature maps. *Cartographic Journal* **10**, 17-21.

Davidoff, J.B. (1987). The role of colour in visual displays. *International Reviews of Ergonomics* **1**, 21-42.

Dickson, G.W., DeSanctis, G., and McBride, D.J. (1986). Understanding the effectiveness of computer graphics for decision support: A cumulative experimental approach. *Communications of the ACM*, Volume 29, Number 1, 40-47.

Dolby, J.L., Clark, N., Rogers, W.H. (1986). The language of data: A general theory of data. In *Proceedings of the Eighteenth Symposium on the Interface.*

Easterby, R.S. (1970). The perception of symbols for machine displays. *Ergonomics* **13** (1), 149-158.

Ehrenberg, A.S.C. (1975). *Data Reduction*. New York: John Wiley & Sons.

Eriksen, C.W. and Hake, H.W. (1955). Multidimensional stimulus differences and accuracy of discrimination. *Journal of Experimental Psychology* **50**, 153-160.

Ericsson, K.A., Chase, W.G., and Faloon, S. (1980). Acquisition of a memory skill. *Science* **208**, 1181-1182.

Fallside, D. (1988). *Understanding Machines in Motion*. PhD Thesis, Department of Psychology, Carnegie-Mellon University.

Farmer, E.W., and Taylor, R.M. (1980). Visual search through color displays: Effects of target-background similarity and background uniformity. *Perception and Psychophysics* **9**, 289-292.

Feiner, S. (1985). APEX: An experiment in the automatic creation of pictorial explanations. *IEEE Computer Graphics and Applications*, May, 29-37.

Feiner, S.K., and McKeown, K.R. (1990). Generating coordinated multimedia explanations. In *Proceedings of the 6th IEEE Conference on Artificial Intelligence Applications*, Santa Barbara, CA, March 5-9.

Fisher, D.L., and Tan, K.C. (1989). Visual displays: The highlighting paradox. *Human Factors* **31** (1), 17-30.

Freund, L.E., and Sadosky, T.L. (1967). Linear programming applied to optimization of instrument panel and workplace layout. *Human Factors* **9**, 295-300.

Frey, A.H., and Eichert, III, E.S. (1978). An evaluation of holograms as training and job aids. *Human Factors* **20** (6), 661-670.

Friedell, M. (1982). Context-sensitive, graphic presentation of information, *Computer Graphics* **16** (3), 181-188.

Flannery, J.J. (1956). The relative effectiveness of some common graduated point symbols in the presentation of quantitative data. PhD Thesis, University of Wisconsin.

Forgy, C.L. (1979). On the Efficient Implementation of Production Systems. Ph.D. Thesis, Department of Computer Science, Carnegie Mellon University.

Fowler, R.L., and Barker, A.S. (1974). Effectiveness of highlighting for retention of text material. *Journal of Applied Psychology* **59** (3), 358-364.

Funkhouser, H.G. (1936). A note on a tenth century graph. *Osiris* **1**, January, 260-262.

Funkhouser, H.G. (1937). Historical development of the graphical representation of statistical data. *Osiris* **3**, p. 326.

Garfinkel, H. (1972). Remarks on ethnomethodology. In J. Gumperz and D. Hymes (Eds.), *Directions in Sociolinguistics: The Ethnography of Communication*. New York: Holt, Reinhart, and Winston.

Gargan, R.A., Sullivan, J.W., and Tyler, S.W. (1988). Multimodal response planning: An adaptive rule-based approach. In *Proceedings of the 1988 Conference on Human Factors in Computing Systems*, Washington, D.C., 229-234.

Garner, W.R. (1974). *The Processing of Information and Structure*. Hillsdale, NJ: Lawrence Erlbaum.

Garner, W.R. and Felfoldy, G.L. (1970). Integrality of stimulus dimensions in various types of information processing. *Cognitive Psychology* **1**, 225-241.

Gibson, R., and Laios, L. (1978). The presentation of information to the job-shop scheduler. *Human Factors, 20 (6)*, 725-732.

Gnanamgari, S. (1981). Information presentation through default displays, PhD dissertation, University of Pennsylvania, May 1981.

Goldenberg, E.P. (1989). Seeing beauty in mathematics: Using fractal geometry to build a spirit of mathematical inquiry. *Journal of Mathematical Behavior* **8**, 169-204.

Goldenberg, E.P. (1988). Mathematics, metaphors, and human factors: Mathematical, technical, and pedagogical challenges in the educational use of graphical representation of functions. *Journal of Mathematical Behavior*, *7*, 135-173.

Goldstein, D.A., and Lamb, J.C. (1967). Visual coding using flashing lights. *Human Factors* **9** (5), 405-408.

Gould, J.D., and Lewis, C.H. (1985). Designing for usability: Key principles and what designers think. *Communications of the ACM, 28 (3),* 300-311.

Gould, J.D., and Schaffer, A. (9167). The effects of divided attention on visual monitoring of multi-channel displays. *Human Factors* **9** (3), 191-202.

Green, B.F., and Anderson, L.K. (1956). Color coding in a visual search task. *Journal of Experimental Psychology* **57**, 19-24.

Grudin, J. (1989). The case against user interface consistency. *Communications of the ACM*, Volume 32, Number 10, 1164-1173.

Hall, R.O. (1943). *Handbook of Tabular Presentation: How To Design and Edit Statistical Tables.* New York: Ronald Press.

Haskell, A.C. (1926). *Graphic Charts in Business.* New York: Codex Book Company.

Hayes, J.R., and Simon, H.A. (1974). Understanding written problem instructions. In L. Gregg (Ed.), *Knowledge and Cognition.* Hillsdale, NJ: Lawrence Erlbaum.

Haygood, R.C., Teel, K.S., and Grenning, C.P. (1964). Link analysis by computer. *Human Factors* **6**, 63-70.

Hegarty, M. and Just, M. (1988). Understanding machines from text and diagram. In H. Mandl and J. Levin (Eds.), *Knowledge Acquisition from Text and Picture,* Amsterdam: North-Holland.

Hendrick, C., Wallace, G. and Tappenbeck, J. (1968). Effects of cognitive set on color perception. *Journal of Personality and Social Psychology* **10**, 487-494.

Hudson, W. (1968). The study of the problem of pictorial perception among accultured groups. *International Journal of Psychology,* 2, 89-107.

Huff, D. (1954). *How To Lie With Statistics.* Norton.

Jacob, R.J.K., Egeth, H.E., and Bevan, W. (1976). The face as a data display. *Human Factors* **18** (2), 189-200.

Jaffe, A.J. and Spirer H.F. (1987). *Misused Statistics: Straight Talk for Twisted Numbers.* New York: Marcel Dekker, Inc.

Jarvenpaa, S.L. and Dickson, G.W. (1988). Graphics and managerial decision making: Research based guidelines. *Communications of the ACM*, Volume 31, Number 6, 764-774.

Jenks, C.F., and Knos, D.S. (1961). The use of shading patterns in graded series. *Annals of the Association of American Geographers* **51**, 316-334.

Julesz, B. (1981). Textons, the elements of texture perception, and their interaction. *Nature* **290**, 91-97.

Kahneman, D., and Henik, A. (1981). Perceptual organization and attention. In M. Kubovy and J. R. Pomerantz (Eds.), *Perceptual Organization*. Hillsdale, NJ: Lawrence Erlbaum, 181-211.

Kammann, R. (1975). The comprehensibility of printed instructions and the flowchart alternative. *Human Factors* **17** (2), 183-191.

Kanarick, A.F., and Petersen, R.C. (1971). Redundant color coding and keeping-track performance. *Human Factors* **13** (2), 183-188.

Karsten, K.G. (1923). *Charts and Graphs: An Introduction to Graphic Methods in the Control and Analysis of Statistics*. New York: Prentice-Hall.

Kieras, D., and Polson, P.G. (1985). An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies, 22*, 365-394.

Klahr, D. (1973). Quantification processes, in W.G. Chase (Ed.), *Visual Information Processing*, New York: Academic Press, 3-34.

Koedinger, K.R. (1989). An example of insight through referent-based reasoning. 1989 Pitt-CMU Graduate Conference on Cognition.

Koedinger, K.R. and Anderson, J. R. (in press). Abstract planning and perceptual chunks: Elements of expertise in geometry, to appear in *Cognitive Science*.

Kosslyn, S.M. (1989). Understanding charts and graphs. *Applied Cognitive Psychology, 3*, 185-226.

Kosslyn, S.M. (1985). Graphics and human information processing: A review of five books. *Journal of the American Statistical Association* **80** (391), 499-512.

Kosslyn, S.M. (1980). *Image and Mind*. Cambridge, MA: Harvard University Press.

Krohn, G.S. (1983). Flowcharts used for procedural instructions. *Human Factors 25 (5)*, 573-581.

Kubovy, M., and Pomerantz, J.R., (Eds.), (1981). *Perceptual Organization*. Hillsdale, NJ: Lawrence Erlbaum.

Larkin, J.H. (1989). Display-Based Problem Solving. In D. Klahr and K. Kotovsky (Eds.), *Complex Information Processing: The Impact of Herbert A. Simon*. Hillsdale, NJ: Lawrence Erlbaum, 319-342.

Larkin, J.H. and Simon, H.A. (1987). Why a diagram is (sometimes) worth 10,000 words. *Cognitive Science, 11*, 65-99.

Larsen, A., and Bundesen, C. (1978). Size scaling in visual pattern recognition. *Journal of Experimental Psychology: Human Perception and Performance* **4** (1), p. 1.

Lewis, C.H. (in press). A research agenda for the nineties in human-computer interaction. *Human-Computer Interaction*.

Lockwood, A. (1969). *Diagrams: A Visual Survey of Graphs, Maps, Charts, and Diagrams for the Graphics Designer*. Watson-Guptill.

Loftus, E.F. (1977). Shifting human color memory. *Memory and Cognition* **5** (6), 696-699.

Lucas, H.C. Jr. (1981). An experimental investigation of the use of computer-based graphics in decision making. *Management Science* **27** (7), 757-768.

Luder, C.B., and Barber, P.J. (1984). Redundant color coding in airborne CRT displays. *Human Factors* **26**, 19-32.

Luria, S.M., and Kinney, J.A.S. (1970). Underwater vision. *Science*, 167, 1454-1461.

Lusk, E.J., and Kersnick, M. (1979). The effect of cognitive style and report format on task performance: The MIS design consequences. *Management Science* **25** (8), 787-798.

Macdonald-Ross, M. (1977). How numbers are shown: A review of research on the presentation of quantitative data in texts. *Audio-Visual Communication Review* **25** (4), 359-409.

Mackinlay, J.D. (1986). Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics, 5 (2)*, 110-141.

Mackinlay, J.D., and Genesereth, M. (1985). Expressiveness and language choice. *Data and Knowledge Engineering, 1*. North Holland, 17-29.

Marey, E.J. (1885). *La Methode Graphique*, Paris.

Marr, D., and Nishihara, H.K. (1978). Representation and recognition of the spatial organization of 3-dimensional shapes. *Proceedings of the Royal Society* **B 200**, 269-294.

Marsh, C. (1988). *Exploring Data.* Cambridge, UK: Polity Press.

Martin, J., and McClure, C. (1985). *Diagramming Techniques for Analysts and Programmers*. Englewood Cliffs, NJ: Prentice-Hall.

Matthews, M.L. (1986). The influence of visual workload history on visual performance. *Human Factors* **28** (6), 623-632.

McCleary, G.F. (1981). How to design an effective graphics presentation. In P.A. Moore (Ed.), *How To Design an Effective Graphics Presentation*, Harvard Library of Computer Graphics, 15-64.

Modley, R. (1932). *How To Use Pictorial Statistics*. New York: Funk and Wagnalls.

Monk, T.H. (1981). The interaction between the edge effect and target conspicuity in visual search. *Human Factors* **23** (5), 615-625.

Monk, T.M. (1976). Target uncertainty in applied visual search. *Human Factors* **18** (6), 607-612.

Monk, T.M., and Brown, B. (1975). The effect of target surround density on visual search performance. *Human Factors* **17** (4), 356-360.

Morawski, T., Drury, C.G., and Karwan, M.H. (1980). Predicting search performance for multiple targets. *Human Factors* **22** (6), 707-718.

Morgan, B.B., Jr. and Alluisi, E.A. (1967). Effects of discriminability and irrelevant information on absolute judgements. *Perception and Psychophysics* **2**, 54-58.

Neal, J.G. and Shapiro, S.C. (1989). Intelligent multi-media interface technology. In J. Sullivan and S. Tyler (Eds.), *Architectures for Intelligent Interfaces: Elements and Prototypes*. Addison-Wesley, Reading, MA.

Neisser, U. (1963). Decision time without reaction time: Experiments in visual scanning. *American Journal of Psychology, 76*, 376-385.

Neisser, U., Novick, R., and Lazar, R. (1963). Searching for ten targets simultaneously. *Perceptual and Motor Skills* **17**, 955-961.

Neisser, U. (1964). Visual search. *Scientific American* **210**, 94-102.

Neurath, O. (1936). *International Picture Language*. London: Kegan Paul.

Neurath, O. (1944). Visual aids and arguing. *New Era* **25**, 51-61.

Newell, A. (1973). You can't play 20 questions with nature and win. In W. G. Chase (Ed.), *Visual Information Processing*. New York: Academic Press.

Newell, A., and Simon, H.A. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.

Nickerson, R.S. (1966). Response times with memory-dependent decision task. *Journal of Experimental Psychology* **72** (5), 761-769.

Norman, D.A. (1988). *The Psychology of Everyday Things*. New York: Basic Books.

Ohlsson, S. (1987). Sense and reference in the design of interactive illustrations for rational numbers. In R. Lawler and M. Yazdani (Eds.), *Artificial Intelligence and Education*. Norwood, NJ: Ablex.

Paivio, A. and te Linde, J. (1980). Symbolic comparisons of objects on colour attributes. *Journal of Experimental Psychology: Human Perception and Performance* **6**, 652-661.

Payne, S., and Green, T.R.G. (1986). Task-action grammars: A model of the mental representation of task languages. *International Journal of Man-Machine Studies, 2*, 93-133.

Playfair, W. (1786). *The Commercial and Political Atlas.*

Playfair, W. (1798). *Lineal Arithmetic.*

Playfair, W. (1801). *Statistical Breviary.*

Pomerantz, J.R., Sager, L.C., and Stoever, R.J. (1977). Perception of wholes and of their component parts: Some configural superiority effects. *Journal of Experimental Psychology: Human Perception and Performance* **3** (3), 422-435.

Pylyshyn, Z. (1981). The imagery debate: Analogue media versus tacit knowledge, *Psychological Review, 88*, 16-45.

Remington, R., and Williams, D. (1986). On the selection and evaluation of visual display symbology: Factors influencing search and identification times. *Human Factors* **28** (4), 407-420.

Robinson, C.P., and Eberts, R.E. (1987). Comparison of speech and pictorial displays in a cockpit environment. *Human Factors, 29 (1)*, 31-44.

Roth, S.F., Mattis, J., and Mesnard, X. (1989). Graphics and natural language as components of automatic explanation. In J. Sullivan and S. Tyler (Eds.), *Architectures for Intelligent Interfaces: Elements and Prototypes*. Addison-Wesley, Reading, MA.

Saenz, N.E., and Riche, Jr., C.V. (1974). Shape and color as dimensions of a visual redundant code. *Human Factors* **16** (3), 308-313.

Salomon, G. (1972). Can we affect cognitive skills through visual media?: An hypothesis and initial findings. *A-V Communication Review, 20 (4)*, 401-422.

Schmid, C.F. (1983). *Statistical Graphics: Design Principles and Practices*. New York: John Wiley and Sons.

Schneider, W. (1985). Training high-performance skills: Fallacies and guidelines. *Human Factors, 27 (3)*, 285-300.

Seligmann, D.D., and Feiner, S.K. (1989). Specifying composite illustrations with communicative goals. In *Proceedings of ACM Symposium on User Interface Software and Technology*, Williamsburg, VA, November 13-15, 1-9.

Shneiderman, B. (1987). *Designing the User Interface*. Reading, MA: Addison-Wesley.

Shontz, W.D., Trumm, G.A., and Williams, L.G. (1971). Color coding for information location. *Human Factors* **13** (3), 237-246.

Simon, H.A. and Barenfeld, M. (1969). Information-processing analysis of perceptual processes in problem solving. *Psychological Review* **76** (5), 473-483.

Smith, S.L., and Goodwin, N.C. (1971). Blink coding for information display. *Human Factors* **13** (3), 283-290.

Smith, S.L., and Goodwin, N.C. (1972). Another look at blinking displays. *Human Factors* **14** (4), 345-347.

Smith, S.L. (1962). Color coding and visual search. *Journal of Experimental Psychology* **64**, 434-440.

Smith, S.L. (1963). Color coding and visual separability in information displays. *Journal of Applied Psychology* **47**, 358-364.

Stevens, S.S. (1946). On the theory of scales of measurement, *Science* **103** (2684), 677-680.

Stevens, S.S. (1974). Perceptual magnitude and its measurement. in E. C. Carterette and M. P. Friedmann (Eds.), *Handbook of Perception* (Vol. 2), New York: Academic Press, 361-389.

Teghtsoonian, J. (1965). The judgement of size. *American Journal of Psychology* **78**, 392-402.

Teichner, W.H., and Krebs, M.J. (1972). Estimating the detectability of target luminances. *Human Factors* **14** (6), 511-519.

Teichner, W.H., and Mocharnuk, J.B. (1979). Visual search for complex targets. *Human Factors* **21** (3), 259-275.

Thagard, P. (1988). Explanatory coherence. *Behavioral and Brain Sciences.*

Treisman, A. (1977). Focused attention in the perception and retrieval of multidimensional stimuli. *Perception and Psychophysics* **22**, 1-11.

Treisman, A., and Gelade, G. (1980). A feature integration theory of attention. *Cognitive Psychology* **12**, 97-136.

Tufte, E.R. (1990). *Envisioning Information.* Graphics Press, Cheshire, Connecticut.

Tufte, E.R. (1983). *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, Connecticut.

Tullis, T.S. (1981). An evaluation of alphanumeric, graphic, and color information displays. *Human Factors* **23** (5), 541-550.

Ullman, J.D. (1980). *Principles of Database Systems.* Computer Science Press.

Ullman, S. (1984). Visual routines. *Cognition* **18**, 97-159.

van Nes, F.L., Juola, J.F., and Moonen, R.J.A. (1987). Facilitation and inhibition of visual display search processes through the use of colour. Institute of Perception Research Annual Progress Report, Eindhoven, The Netherlands.

Vartabedian, A.G. (1971). The effects of letter size, case, and generation method on CRT display search time. *Human Factors* **13** (4), 363-368.

Vernoy, M.W. (1989). Simultaneous adaptation to size, distance, and curvature underwater. *Human Factors* **31**(1), 77-85.

Ware, C., and Beatty, J.C. (1988). Using color dimensions to display data dimensions. *Human Factors* **30** (2), 127-142.

Washburne, J.N. (1927). An experimental study of various graphic, tabular, and textual methods of presenting quantitative material. *Journal of Educationa' Psychology* **18** (6), 361-376, 465-477.

Wedell, J. and Alden, D.G. (1973). Color versus numeric coding in a keep-track task: Performance under varying load conditions. *Journal of Applied Psychology* **57**, 154-159.

Wickens, C.D. and Goettle, B. (1984). Multiple resources and display formatting: The implications of task integration. In *Proceedings of the Human Factors Society - 28th Annual Meeting.*

Wickens, C.D., Sandry, D.L., and Vidulich, M. (1983). Compatibility and resource competition between modalities of input, central processing, and output. *Human Factors* **25**(2), 227-248.

Williams, R.L. (1956). *Statistical Symbols for Maps: Their Design and Relative Values*. New Haven, CT: Yale University, Map Laboratory.

Witkin, H., Oltman, P.K., Ruskin, E., and Karp, S.A. (1971). *The Embedded Figures Test*. Palo Alto, CA: Consulting Psychologist Press.

Wright, P. (1968). Using tabulated information. *Ergonomics* **11**, 331-343.

Wright, P. (1972). Explicit and implicit tabulation formats. *Ergonomics* **15**, 175-187.

Zeki, S.M. (1978). Functional specialization in the visual cortex of the rhesus monkey. *Nature* **274**, 423-428.

Zipf, G.K. (1935). *The Psychobiology of Language*. Boston: Houghton-Mifflin.

Zdybel, F., Greenfield, N.R., Yonke, M.D., and Gibbons, J. (1981). An information presentation system, in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, 978-984.

Zwaga, H.J.G. and Duijnhouwer, F. (1984). The influence of a fixed background in a VDU display on the efficiency of color and shape coding. In *Color Coded vs. Monochrome Electronic Displays*, Proceedings of a NATO Workshop, C.P. Gibson (Ed.), London: HMSO.